

CE

ecomat mobile

Original-Programmierhandbuch
ExtendedSafetyController

ecomat100

CR7132

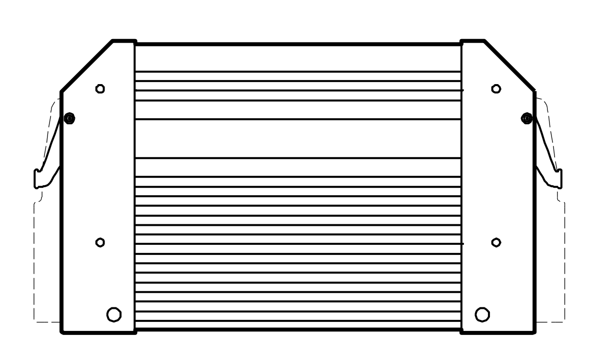
für ISO 13849 bis PL d
für IEC 62061 bis SIL CL 2

Laufzeitsystem V01.00.05

CODESYS® ≥ v2.3.9.42 (< v3.0)

Deutsch

7390679_09_DE 2014-12-15



Inhaltsverzeichnis

1 Über diese Anleitung

Inhalt

2026088

1.1 Übersicht: Dokumentations-Module für Safety-ecomatmobile-Geräte

14401

Die Dokumentation für **ecomat*mobile***-Geräte besteht aus folgenden Modulen:

1.	**Datenblatt**
Inhalt:	Technische Daten in Tabellenform
Quelle:	www.ifm.com > Land wählen > [Datenblattsuche] > CR7132 > [Technische Daten im PDF-Format]
2.	**Montageanleitung / Betriebsanleitung**
Inhalt:	Anleitung für Montage, elektrische Installation, (Inbetriebnahme*), Technische Daten
Quelle:	Anleitung wird mit dem Gerät mitgeliefert Auch zu finden auf der ifm-Homepage: www.ifm.com > Land wählen > [Datenblattsuche] > CR7132 > [Betriebsanleitungen]
3.	**Programmierhandbuch + Online-Hilfe**
Inhalt:	Beschreibung der Konfiguration und der Funktionen der Geräte-Software
Quelle:	www.ifm.com > Land wählen > [Datenblattsuche] > CR7132 > [Software Download] > [Target and Documentation Package] > den Anweisungen folgen

4.	Systemhandbuch "Know-How ecomatmobile"
Inhalt:	Hintergrundwissen zu folgenden Themen: • Übersicht Templates und Demo-Programme • CAN, CANopen • Ausgänge steuern • User-Flash-Speicher • Visualisierungen • Übersicht Dateien und Bibliotheken
Quelle:	www.ifm.com > Land wählen > [Datenblattsuche] > CR7132 > [Betriebsanleitungen]

*) Die in Klammern gesetzten Beschreibungen sind nur in den Anleitungen bestimmter Geräte enthalten.

1.2 CODESYS-Programmierhandbuch

17542

Im ergänzenden "Programmierhandbuch CODESYS V2.3" der 3S GmbH erhalten Sie weitergehende Informationen über die Nutzung des Programmiersystems.
Dieses Handbuch steht auf der **ifm**-Homepage als kostenloser Download zur Verfügung:
→ www.ifm.com > [Service] > [Download] > [Systeme für mobile Arbeitsmaschinen]
Handbücher und Online-Hilfen für **ecomat*mobile*** finden Sie auch hier:
→ **ecomat*mobile***-DVD "Software, tools and documentation"

1.3 Was bedeuten die Symbole und Formatierungen?

203

Folgende Symbole oder Piktogramme verdeutlichen Ihnen unsere Hinweise in unseren Anleitungen:

⚠ WARNUNG
Tod oder schwere irreversible Verletzungen sind möglich.

⚠ VORSICHT
Leichte reversible Verletzungen sind möglich.

ACHTUNG
Sachschaden ist zu erwarten oder möglich.

[!]	Wichtige Hinweise auf Fehlfunktionen oder Störungen
[i]	Weitere Hinweise
► ...	Handlungsaufforderung
> ...	Reaktion, Ergebnis
→ ...	"siehe"
abc	Querverweis
123 0x123 0b010	Dezimalzahl Hexadezimalzahl Binärzahl
[...]	Bezeichnung von Tasten, Schaltflächen oder Anzeigen

1.4 Wie ist diese Dokumentation aufgebaut?

204
1508

Diese Dokumentation ist eine Kombination aus verschiedenen Anleitungstypen. Sie ist eine Lernanleitung für den Einsteiger, aber gleichzeitig auch eine Nachschlageanleitung für den versierten Anwender. Dieses Dokument richtet sich an die Programmierer der Anwendungen.

Und so finden Sie sich zurecht:

- Um gezielt zu einem bestimmten Thema zu gelangen, benutzen Sie bitte das Inhaltsverzeichnis.
- Mit dem Stichwortregister "Index" gelangen Sie ebenfalls schnell zu einem gesuchten Begriff.
- Am Anfang eines Kapitels geben wir Ihnen eine kurze Übersicht über dessen Inhalt.
- Abkürzungen und Fachbegriffe → Anhang.

Bei Fehlfunktionen oder Unklarheiten setzen Sie sich bitte mit dem Hersteller in Verbindung:
→ www.ifm.com > Land wählen > [Kontakt].

Wir wollen immer besser werden! Jeder eigenständige Abschnitt enthält in der rechten oberen Ecke eine Identifikationsnummer. Wenn Sie uns über Unstimmigkeiten unterrichten wollen, dann nennen Sie uns bitte diese Nummer zusammen mit Titel und Sprache dieser Dokumentation. Vielen Dank für Ihre Unterstützung!

Im Übrigen behalten wir uns Änderungen vor, so dass sich Abweichungen vom Inhalt der vorliegenden Dokumentation ergeben können. Die aktuelle Version finden Sie auf der **ifm**-Homepage:
→ www.ifm.com > Land wählen > [Datenblattsuche] > (Artikel-Nr.) > [Betriebsanleitungen]

1.5 Historie der Anleitung (CR7n32)

9196

Was hat sich wann in dieser Anleitung geändert? Ein Überblick:

Datum	Thema	Änderung
2013-11-04	Beschreibungen der FBs SF_ANTIVALENT und SF_EQUIVALENT	Hinweis zum OSSD ergänzt
2013-11-04	Beispiel aus EN280	Grafik verbessert
2013-11-12	CODESYS-Version	Mindestversion = V2.3.9.42
2013-11-12	Kapitel "Beachten"	Hinweise auf externe Dokumente ergänzt: • "Release Notes" und • "Wichtige Hinweise zum CR7n32"
2014-03-13	FB SAFETY_SWITCH	Ausgangssignal: Werte für Periodendauer
2014-03-13	FB OUTPUT_BRIDGE	Mindestwert für CHANGEOVER_TIME
2014-03-13	FB J1939_X_RECEIVE	Vorgabe: Anzahl empfangener Daten prüfen
2014-03-18	FB SAFETY_SWITCH	Angaben zum FB-Ausgang ERROR
2014-04-28	diverse FBs	Beschreibung FB-Eingang CHANNEL präzisiert
2014-06-24	FB PID2	Grafik korrigiert
2014-06-30	Name der Dokumentation	"Systemhandbuch" umbenannt zu "Programmierhandbuch"
2014-07-31	Kapitel "Fehler-Codes, Diagnose"	Fehler-Code-Tabellen optimiert
2014-07-31	FB PHASE	Beschreibung Parameter der Ausgänge C, ET korrigiert
2014-07-31	FB OUTPUT_CURRENT_CONTROL	Wenn Sollwert=0 mA >> Regelung auf 0 "innerhalb von 100 ms" anstatt "sofort"
2014-08-26	Beschreibung Eingänge, Ausgänge	highside / lowside ersetzt durch plusschaltend / minusschaltend
2014-09-29	Begriff OUT_OVERLOAD_PROTECTION	allgemein ersetzt durch Überlastschutz
2014-10-01	Laufzeitsystem V01.00.04	Abweichung vom zulässigen Wertebereich kann zu schwerem Fehler führen
2014-10-08	Checkliste: Bootprojekt erzeugen	falsch: "Datei in Steuerung schreiben" richtig: "Projekt in Steuerung laden"
2014-11-12	Kapitel "Ausgänge (Technologie)"	Abschnitt "Diagnose der binären Ausgänge" ergänzt oder korrigiert
2014-12-09	Laufzeitsystem v01.00.05	• Downloader Version: min. V6.18.26 • CODESYS-Kommunikation via USB: Baudrate < 115 200 Baud

2 Sicherheitshinweise

Inhalt

213

2.1 Beachten!

13918
11212

Mit den in dieser Anleitung gegebenen Informationen, Hinweisen und Beispielen werden keine Eigenschaften zugesichert. Die abgebildeten Zeichnungen, Darstellungen und Beispiele enthalten weder Systemverantwortung noch anwendungsspezifische Besonderheiten.

► Die Sicherheit der Maschine/Anlage muss auf jeden Fall eigenverantwortlich durch den Hersteller der Maschine/Anlage gewährleistet werden.

► Beachten Sie die nationalen Vorschriften des Landes, in welchem die Maschine/Anlage in Verkehr gebracht werden soll!

⚠ WARNUNG

Bei Nichtbeachten der Hinweise in dieser Anleitung sind Sach- oder Körperschäden möglich!
Die **ifm electronic gmbh** übernimmt hierfür keine Haftung.

► Die handelnde Person muss vor allen Arbeiten an und mit diesem Gerät die Sicherheitshinweise und die betreffenden Kapitel dieser Anleitung gelesen und verstanden haben.

► Die handelnde Person muss zu Arbeiten an der Maschine/Anlage autorisiert sein.

► Die handelnde Person muss für die auszuführende Arbeit über die erforderliche Ausbildung und Qualifikation verfügen.

► Beachten Sie die Technischen Daten der betroffenen Geräte!
Das aktuelle Datenblatt finden Sie auf der **ifm**-Homepage:
→ www.ifm.com > Land wählen > [Datenblattsuche] > (Artikel-Nr.) > [Technische Daten im PDF-Format]

► Beachten Sie die Montage- und Anschlussbedingungen sowie die bestimmungsgemäße Verwendung der betroffenen Geräte!
→ mitgelieferte Montageanleitung oder auf der **ifm**-Homepage:
→ www.ifm.com > Land wählen > [Datenblattsuche] > (Artikel-Nr.) > [Betriebsanleitungen]

► Beachten Sie die Korrekturen und Hinweise in den "Release-Notes" zur vorhandenen Hardware, Software und Dokumentation auf der **ifm**-Homepage:
→ www.ifm.com > Land wählen > [Datenblattsuche] > (Artikel-Nr.) > [Betriebsanleitungen]

15742

⚠ WARNUNG

Bei Nichtbeachten folgender Hinweise sind Sach- oder Körperschäden möglich!
Die **ifm electronic gmbh** übernimmt hierfür keine Haftung.

► Beachten Sie unbedingt auch das Dokument "Wichtige Hinweise zum CR7n32" für die von Ihnen verwendeten Softwarestände!

5020

ACHTUNG

Der Treiberbaustein der seriellen Schnittstelle kann beschädigt werden!

Beim Trennen oder Verbinden der seriellen Schnittstelle unter Spannung kann es zu undefinierten Zuständen kommen, die zu einer Schädigung des Treiberbausteins führen.

► Die serielle Schnittstelle nur im spannungslosen Zustand trennen oder verbinden!

2.2 Welche Vorkenntnisse sind notwendig?

215

Das Dokument richtet sich an Personen, die über Kenntnisse der Steuerungstechnik und SPS-Programmierkenntnisse mit IEC 61131-3 verfügen.

Zum Programmieren der SPS sollten die Personen zusätzlich mit der Software CODESYS vertraut sein.

Das Dokument richtet sich an Fachkräfte. Dabei handelt es sich um Personen, die aufgrund ihrer einschlägigen Ausbildung und ihrer Erfahrung befähigt sind, Risiken zu erkennen und mögliche Gefährdungen zu vermeiden, die der Betrieb oder die Instandhaltung eines Produkts verursachen kann. Das Dokument enthält Angaben zum korrekten Umgang mit dem Produkt.

Lesen Sie dieses Dokument vor dem Einsatz, damit Sie mit Einsatzbedingungen, Installation und Betrieb vertraut werden. Bewahren Sie das Dokument während der gesamten Einsatzdauer des Gerätes auf.

Befolgen Sie die Sicherheitshinweise.

2.3 Anlaufverhalten der Steuerung

13919

⚠ WARNUNG

Gefahr durch unbeabsichtigtes und gefährliches Anlaufen von Maschinen- oder Anlagenteilen!

► Der Programmierer muss bei der Programmerstellung verhindern, dass nach Auftreten eines Fehlers (z.B. NOT-HALT) und der anschließenden Fehlerbeseitigung unbeabsichtigt Maschinen- oder Anlagenteile gefährlich anlaufen können!
⇨ Wiederanlaufsperre realisieren!

► Dazu im Fehlerfall die in Frage kommenden Ausgänge im Programm logisch abschalten!

Ein Wiederanlauf kann z.B. verursacht werden durch:
- Spannungswiederkehr nach Spannungsausfall
- Fehlerbeseitigung nach NOT-HALT

So erreichen Sie sicheres Verhalten der Steuerung:

► Spannungsversorgung im Anwendungsprogramm überwachen.

► Im Fehlerfall alle relevanten Ausgänge im Anwendungsprogramm ausschalten.

► Aktuatoren, die zu gefahrbringenden Bewegungen führen können, zusätzlich im Anwendungsprogramm überwachen (Feedback).

3 Hinweise für sicherheitsrelevante Anwendungen

Inhalt

3750

3.1 Sichere Maschinen mit dem ecomatmobile-SafetyController

13258

Der **ecomat*mobile***-SafetyController ist für den Einsatz in mobilen Arbeitsmaschinen vorgesehen, die Anforderungen an die Funktionale Sicherheit erfüllen müssen. Daher sind die nachfolgenden Erklärungen auf solche sicheren Maschinen begrenzt.

Grundsätzlich gelten die nachfolgenden Beschreibungen sowohl für die mechanischen als auch für die elektrischen und elektronischen Baugruppen und Baugruppenteile der Arbeitsmaschine. Da es hier aber vorrangig um die elektronische Mobilsteuerung "**ecomat*mobile***-SafetyController CR7n32" und ihre Anwendung in Verbindung mit der Maschinenrichtlinie 2006/42/EG geht, konzentrieren wir uns an dieser Stelle auf deren Anwendung als Sicherheitsbauteil nach ISO 13849.

Die bei der Zertifizierung der Mobilsteuerung "**ecomat*mobile***-SafetyController CR7n32" ebenfalls angewandten Normen IEC 62061 oder IEC 61508 werden an dieser Stelle nicht näher betrachtet. Diese Normen beschränken sich ausschließlich auf elektrische, elektronische und programmierbare elektronische Systeme. Diese Normen kommen in der Regel nur zum Einsatz, wenn die für den Maschinenbauer leichter handhabbare Norm ISO 13849 nicht zum gewünschten Ziel führt.

Die ISO 13849 hat den großen Vorteil, dass die Sicherheitsfunktion vom Sensor über die Verarbeitungseinheit (z.B. den **ecomat*mobile***-SafetyController) bis hin zum Aktuator (z.B. Ventil) verschiedene Technologien beinhalten kann.

Das Dokument "Hinweise für sicherheitsrelevante Anwendungen" ist so strukturiert, dass die einzelnen Abschnitte fortlaufend abgearbeitet werden können, um alle wichtigen Schritte bei der Entwicklung der Anwendung für eine sichere Maschinensteuerung zu berücksichtigen.

Zu Anfang eines Kapitels stellen wir innerhalb eines Kastens alle wichtigen Punkte zusammengefasst vor. Anschließend folgt die ausführliche Beschreibung dieser Punkte.

> ! Jeder Anwendungsentwickler ist zu Folgendem verpflichtet:
> • seine mobile Maschine im Einzelnen zu bewerten,
> • die Besonderheiten der jeweiligen Anwendung zu beachten und
> • die relevanten Normen zu kennen und zu beachten.

ifm electronic ist nicht verantwortlich für die korrekte Umsetzung der normativen Anforderungen in einer Anwendung und die Normeninterpretation.

Der Zustand einer Maschine gilt als sicher, wenn von ihr keine →Gefährdung mehr ausgeht. Dies ist meist der Fall, wenn alle gefahrbringenden Bewegungsmöglichkeiten abgeschaltet sind und nicht unerwartet wieder anlaufen können.

3.1.1 Was ist Maschinensicherheit?

13270

Zusammenfassung

Während der Lebensdauer der Maschine darf bei der Ausführung einer Funktion keine Gefährdung ausgehen, die zu einer Verletzung führen kann.

► Eine Risikobeurteilung durchführen für folgende Fälle:
  - Betreiben der Maschine,
  - Einrichten, Montieren oder Warten der Maschine,
  - vorhersehbare Fehlanwendungen an der Maschine oder von speziellen Maschinenfunktionen
  - mögliche Manipulation der Sicherheitsfunktion durch den Endanwender (z.B. Maschinenführer)

► Der Schutz vor Gefährdungen liegt in der Verantwortung des Anwendungs- oder Maschinenherstellers.

Seit Ende 2009 ist in Europa die Anwendung der überarbeiteten Maschinenrichtlinie 2006/42/EG gültig. Die Maschinenrichtlinie muss daher beachtet und angewendet werden, wenn eine mobile Arbeitsmaschine in einem der folgenden Länder eingesetzt werden soll:

- im Europäischen Wirtschaftsraum
  (= in einem Staat der europäischen Gemeinschaft (EU) oder Norwegen, Island, Liechtenstein)
- Türkei
- Schweiz.

Die Maschinenrichtlinie definiert die Sicherheit wie folgt:
Während der Lebensdauer der Maschine darf bei der Ausführung einer Funktion keine Gefährdung ausgehen, die zu einer Verletzung führen kann. Dies gilt für folgende Fälle:

- Betreiben der Maschine,
- Einrichten, Montieren oder Warten der Maschine,
- vorhersehbare Fehlanwendungen an der Maschine oder von speziellen Maschinenfunktionen, insbesondere Schutz vor Manipulation durch den Endanwender (z.B. Maschinenführer).

► Manipulationen verhindern!
Maschine vor unerlaubtem Zugriff über die Service-Schnittstellen schützen. Das gilt in besonderem Maße bei der Realisierung einer Fernwartungsmöglichkeit.

Der Schutz vor Gefährdungen liegt in der Verantwortung des Anwendungs- oder Maschinenherstellers. Dazu muss der Hersteller einer mobilen Arbeitsmaschine eine Risikobeurteilung durchführen!

→ Kapitel ***Risikobeurteilung*** (→ Seite 23)

3.1.2 Anwenden von (Produkt-)Normen

13271

Zusammenfassung

▶ Prüfen, ob Produktnormen (Typ C-Normen) für die mobile Arbeitsmaschine vorliegen.

▶ Wenn vorhanden: Produktnorm anwenden.
  - Welche Anforderung wird an den Performance Level (PL) der Sicherheitsteuerung und an die damit umgesetzte Funktion gestellt?
  - Welche Anforderung wird an die Kategorie (Cat.) der Maschinenfunktion gestellt?
  - Auf welche anderen Normen wird verwiesen?

▶ Falls eine passende Produktnorm fehlt: die in den folgenden Schritten beschriebene Risikobeurteilung durchführen.

Zur Umsetzung der Anforderungen an die Maschinensicherheit kann der Maschinenkonstrukteur auf Normen zurückgreifen, die ihn bei der Erstellung und Herstellung unterstützen: Sicherheitsnormen für Maschinen.

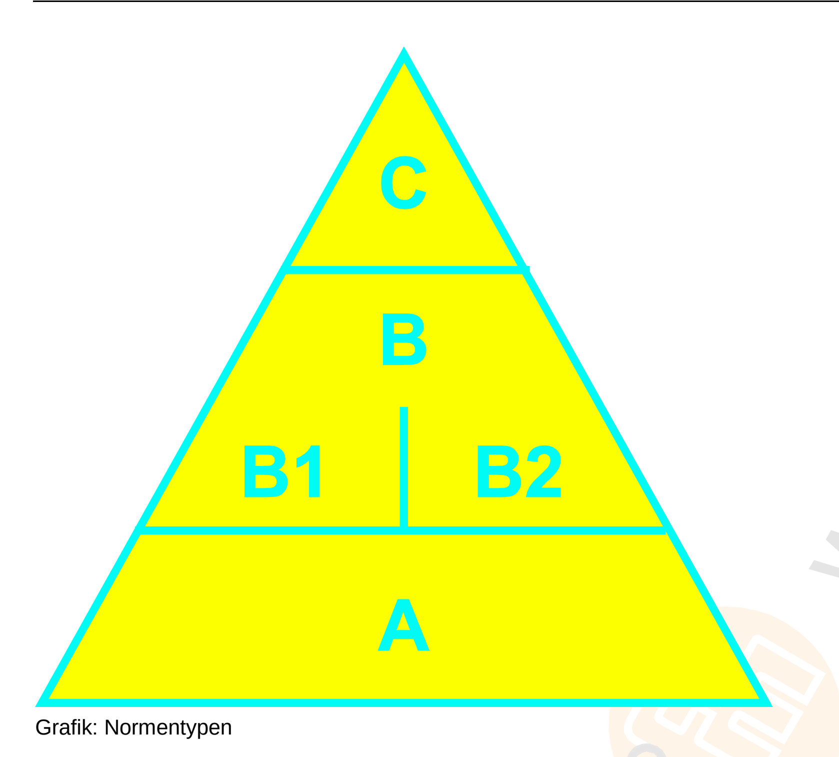
Sicherheitsnormen auf dem Gebiet der Maschinen sind wie folgt strukturiert:
• Typ A-Normen (Sicherheits-Grundnormen)
• Typ B-Normen (Sicherheits-Fachgrundnormen)
• Typ C-Normen (Maschinensicherheitsnormen)

Typ-A-Normen (Sicherheits-Grundnormen) behandeln Grundbegriffe, Entwurfsleitsätze und allgemeine Aspekte, die auf Maschinen angewendet werden können. Beispiele:
• Terminologie, Methodik (ISO 12100),
• Technische Prinzipien (ISO 12100),
• Risikobeurteilung (ISO 12100), ...

Typ-B-Normen (Sicherheits-Fachgrundnormen) behandeln einen Sicherheitsaspekt oder eine Art von Schutzeinrichtungen, die für eine Reihe von Maschinen verwendet werden können. Die Umsetzung auf die konkrete Maschinenfunktion muss bei der Risikobeurteilung, der Spezifikation und der Entwicklung erfolgen. Die Verantwortung für die korrekte Einstufung liegt beim Hersteller der Maschine.

- Typ-B1-Normen für bestimmte Sicherheitsaspekte. Beispiele:
  • Sicherheitsabstände (ISO 13857),
  • Arm-/Hand-Geschwindigkeiten (ISO 13855),
  • Sicherheitsbezogene Teile von Steuerungen (→ISO 13849),
  • Temperaturen, Lärm, ...
- Typ-B2-Normen für Schutzeinrichtungen. Beispiele:
  • NOT-HALT-Schaltungen ((ISO 13850),
  • Zweihand-Schaltungen,
  • trennende oder berührungslos wirkende Schutzeinrichtungen (IEC 61496), ...

Typ-C-Normen (Maschinensicherheitsnormen, Produktnormen) behandeln detaillierte Sicherheitsanforderungen an eine bestimmte Maschine oder eine Gruppe von Maschinen. Oftmals ist dort bereits die Risikoabschätzung für die gesamte Maschine oder Teilfunktionen davon dokumentiert. Bei der Entwicklung einer Maschine sollte man sich deshalb möglichst daran halten. Beispiele:
• Müllfahrzeuge (EN 1501),
• Hubarbeitsbühnen (EN 280), ...



Grafik: Normentypen

Bei der Entwicklung einer mobilen Arbeitsmaschine sollte also zunächst geprüft werden, ob eine Produktnorm, also eine Typ C-Norm, vorhanden ist. In dieser Norm werden ganz konkrete und klare Umsetzungsempfehlungen für die mobile Arbeitsmaschine und Ihre Sicherheitsfunktionen gegeben. Oft werden hier auch klare Aussagen zu den benötigten Sicherheitsstufen (PL = Performance Level) und Sicherheitsstrukturen (Cat = Kategorie) gemacht.

Eine Risikobeurteilung und Risikoabschätzung ist dann nicht mehr erforderlich.

HINWEIS

► Genau prüfen, welche Maschinenfunktionen als sicherheitskritisch einzustufen sind!

Werden mehr Maschinenfunktionen als sicherheitskritisch eingestuft, als notwendig wären, können folgende Nachteile eintreten:
- die Verfügbarkeit der mobilen Arbeitsmaschine wird abnehmen
- die Bedienbarkeit der Maschine wird abnehmen
- die Benutzer der Maschine werden zunehmend unzufrieden
- die Benutzer versuchen, "hindernde" Sicherheitseinrichtungen außer Kraft zu setzen
- die mobile Arbeitsmaschine wird dadurch in letzter Konsequenz unsicherer.

► Also: bei allen nachfolgend beschriebenen Arbeitsschritten die tatsächlich erforderliche Sicherheitseinstufung berücksichtigen!

3.2 Empfohlene Schritte zu einer sicheren Maschine

Inhalt

13267

Nachfolgend beschriebene Schritte haben sich bewährt auf dem Weg zu einer sicheren Maschine.

Beim Maschinenbauer liegt die Gesamtverantwortung für ...
- das Maschinenkonzept,
- die Identifikation der Sicherheitsfunktionen und das Sicherheitskonzept
- die damit verbundenen Kennwerte (PL und Kategorie),
- die Auswahl der Komponenten und
- die Überprüfung (Validierung) der Sicherheitsfunktionen.

► Diesen gesamten Prozess nachvollziehbar dokumentieren und archivieren!

3.2.1 Erstellen des Sicherheitskonzepts und die Risikobeurteilung

Inhalt

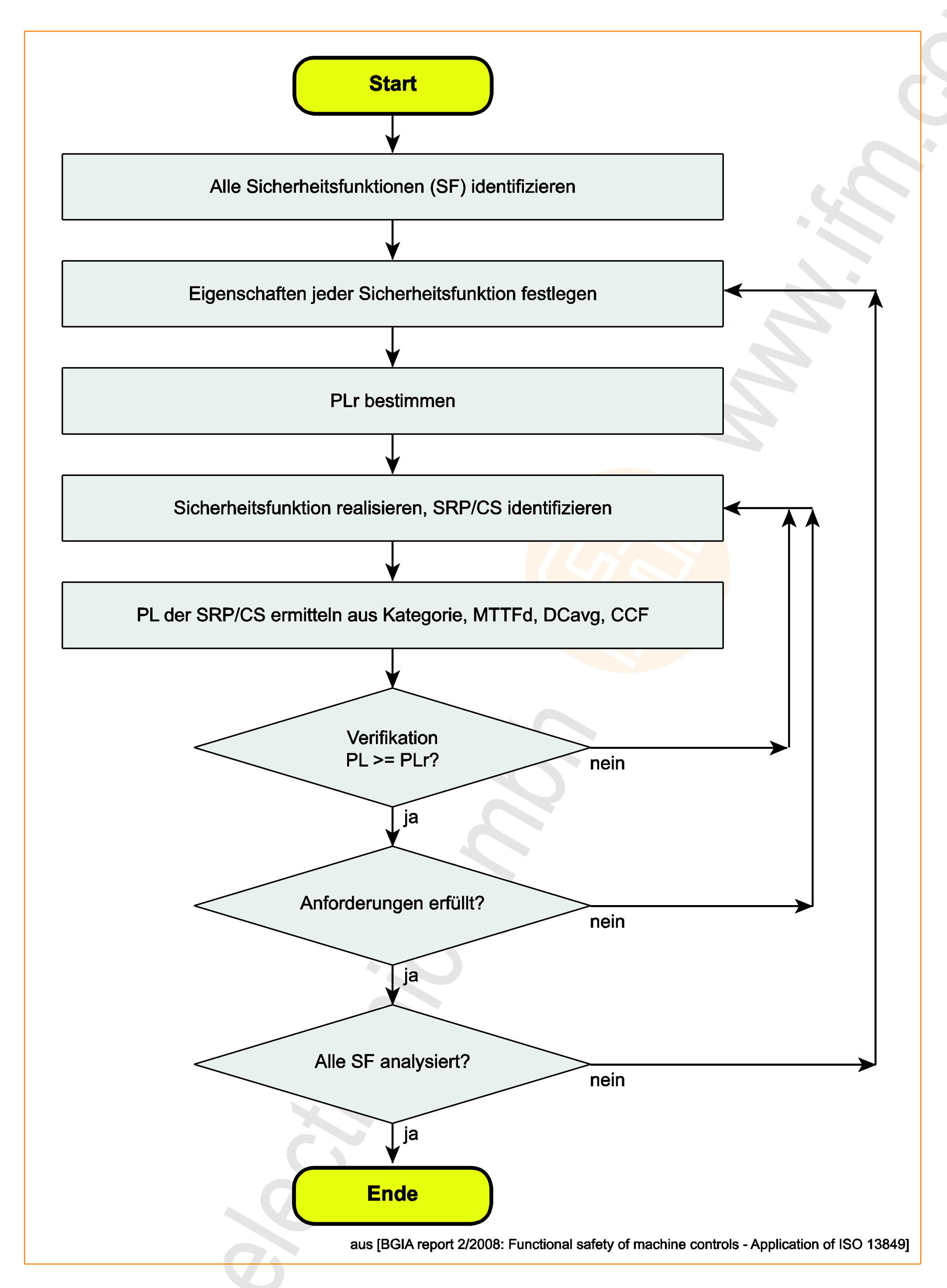
13273

Zusammenfassung

- Prüfen, ob Produktnormen (Typ C-Normen) für die mobile Arbeitsmaschine vorliegen! Falls ja, kann ein Teil der nächsten Punkte übersprungen werden.
- Räumliche Grenzen, den Anwendungszeitraum und die geforderte Verwendung der mobilen Maschine festlegen!
- Mögliche Gefährdungen identifizieren!
- Notwendige und geeignete Sicherheitsfunktionen festlegen!
- Die Risikobeurteilung für jede einzelne Sicherheitsfunktion getrennt durchführen!
- Restrisiko für die mobile Arbeitsmaschine beurteilen auf Basis der Risikoanalyse und Risikobewertung!
- Die Entscheidungswege der Risikobeurteilung klar dokumentieren und archivieren!

Prozess der Risikominderung nach ISO 12100

14329



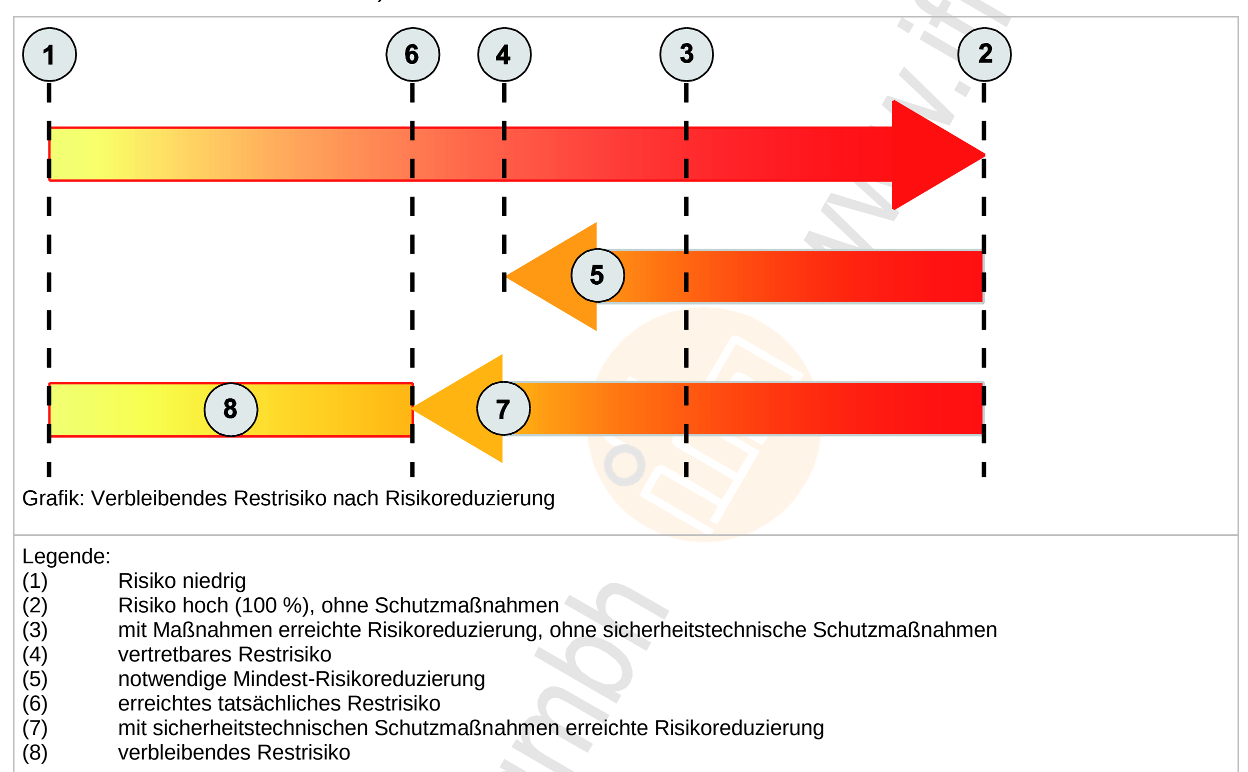
aus [BGIA report 2/2008: Functional safety of machine controls - Application of ISO 13849]

Risiko reduzieren

13259

Grundsätzlich gibt es bei technischen Einrichtungen kein Null-Risiko. Das verbleibende Restrisiko muss auf ein akzeptables Maß reduziert werden. Nur dieses akzeptable Restrisiko wird vom Bediener der Maschine und den Personen im Umfeld getragen.

Die nachfolgende Grafik verdeutlicht die von der Norm EN 13849 definierten Risikoelemente (hier: an einer mobilen Arbeitsmaschine):



Grafik: Verbleibendes Restrisiko nach Risikoreduzierung

Legende:
(1) Risiko niedrig
(2) Risiko hoch (100 %), ohne Schutzmaßnahmen
(3) mit Maßnahmen erreichte Risikoreduzierung, ohne sicherheitstechnische Schutzmaßnahmen
(4) vertretbares Restrisiko
(5) notwendige Mindest-Risikoreduzierung
(6) erreichtes tatsächliches Restrisiko
(7) mit sicherheitstechnischen Schutzmaßnahmen erreichte Risikoreduzierung
(8) verbleibendes Restrisiko

Ist das tatsächliche Restrisiko geringer als das vertretbare Restrisiko, dann ist die geplante sicherheitstechnische Schutzmaßnahme ausreichend und die Maschine darf so konstruiert und aufgebaut werden.

► Den Betreiber der Maschine über das verbleibende Restrisiko informieren:
  • mittels (zwingend) notwendiger Maschinendokumentation und
  • gegebenenfalls zusätzlicher Warnhinweise an der Maschine.

Normen zur Bestimmung und zur Minderung des Risikos:
• aus ISO 13849 wird der Risikograf angewendet
• ISO 12100 beschreibt verschiedene Verfahren zur Bestimmung des Risikos

Sicherheitskonzept

13420

► Ein Sicherheitkonzept erstellen, um das Risiko zu reduzieren, das von einer mobilen Arbeitsmaschine ausgeht!

Falls eine Produktnorm (Typ-C-Norm) für diesen Maschinentyp vorliegt, können aus ihr die einzelnen Sicherheitsfunktionen und die vorgesehene Umsetzung entnommen werden.

► Alle vorgesehenen Sicherheits- und Maschinenfunktionen für den konkreten Einsatzfall überprüfen und – wenn notwendig – anpassen.

► Falls keine Produktnorm vorliegt:
Der Konstrukteur muss die einzelnen Sicherheitsfunktionen mittels Risikoanalyse selbst ermitteln.

► Soweit möglich, die Maschinensicherheit wie folgt realisieren:
 • durch konstruktive Maßnahmen,
 • mit dem Anwendungsprogramm und
 • mit technischen Schutzeinrichtungen.

Risikobeurteilung

13280

Das ist die Gesamtheit des Verfahrens, das die →Risikoanalyse und die →Risikobewertung umfasst. Nach Maschinenrichtlinie 2006/42/EG gilt: "Der Hersteller einer Maschine oder sein Bevollmächtigter hat dafür zu sorgen, dass eine Risikobeurteilung vorgenommen wird, um die für die Maschine geltenden Sicherheits- und Gesundheitsanforderungen zu ermitteln. Die Maschine muss dann unter Berücksichtigung der Ergebnisse der Risikobeurteilung konstruiert und gebaut werden." (→ Anhang 1, Allgemeine Grundsätze)

Risikoanalyse

13278

Kombination aus ...

- Festlegung der Grenzen der Maschine (Verwendungszweck, zeitliche Grenzen),
- Identifizierung der →Gefährdung (Eingreifen von Personen, Betriebszustände der Maschine, vorhersehbarer Missbrauch) und
- der Risikoeinschätzung (Verletzungsgrad, Schadensumfang, Häufigkeit und Dauer der Gefahr, Eintrittswahrscheinlichkeit, Möglichkeit zur Vermeidung oder Begrenzung des →Schadens).

Kurz:
 • **Wo** in und an der Maschine drohen **welche** Gefahren?
 • in **welchen** Situationen?
 • **wie oft** und **wie lange** können diese Gefahren auftreten?
 • **wie groß** wäre der auftretende Schaden?

Risikobewertung

13279

Das ist die auf der →Risikoanalyse beruhende Beurteilung, ob die Ziele zur Risikominderung erreicht wurden.

► Bei der Risikobewertung auch die Erfahrungen berücksichtigen, die aus Unfällen bei der Anwendung vergleichbarer mobiler Arbeitsmaschinen stammen, soweit solche Erfahrungen vorliegen!

Sicherheitsfunktionen festlegen

13277

Welche Sicherheitsfunktionen benötigt werden, hängt ab...
- von der Anwendung der mobilen Arbeitsmaschine,
- von der Gefährdung, die von der Maschine ausgeht.

Die Norm ISO 13849 ist unabhängig von der Anwendung und der verwendeten Technik (z.B. Steuerung oder Hydraulik). In der Norm werden typische Sicherheitsfunktionen aufgeführt, z.B.:
- lokale Steuerungsfunktionen im Gefahrenbereich
- Mutingfunktionen
- Zustimmfunktionen
- unerwarteten Anlauf verhindern
- Befreiung und Rettung eingeschlossener Personen
- Steuerungsfunktionen und Betriebsarten wählen
- Maschine im Notfall stillsetzen

Neben der Auswahl der Sicherheitsfunktionen müssen auch ihre Eigenschaften in den verschieden Betriebsbedingungen der mobilen Arbeitsmaschine betrachtet werden. Dazu zählen z.B.:
- Einsatz bei verschiedenen Betriebsarten, z.B.:
  - Automatikbetrieb,
  - manueller Betrieb,
  - Beseitigung einer Störung
- Reaktion der Maschine beim Ansprechen der Sicherheitseinrichtung
- Ansprechzeit der Sicherheitsfunktion
- Reaktion der Maschine beim Erkennen eines Fehlers in der Sicherheitsfunktion
- Häufigkeit der Betätigung der Sicherheitsfunktion

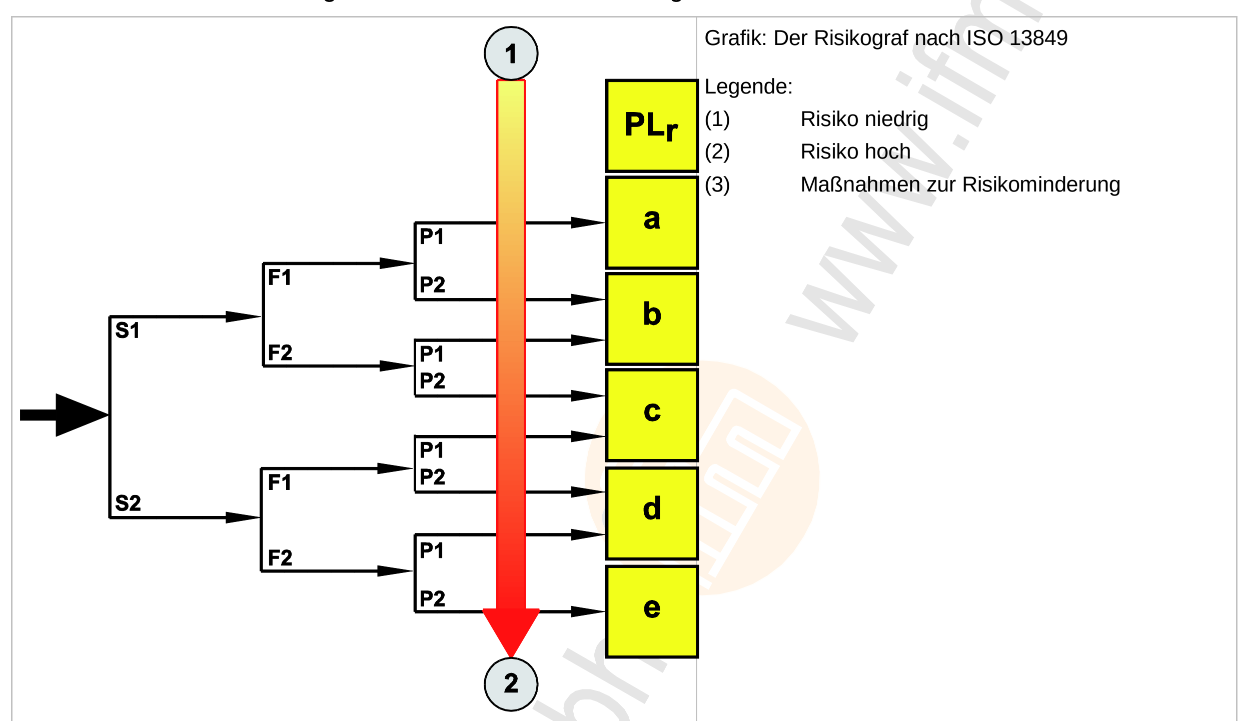
Erforderlichen PL (=PLr) mittels Risikograf herleiten

13275

Beim Risikografen wird die Risikobewertung nach folgender vereinfachter Formel ermittelt:

Risiko = Schadenausmaß x Eintrittswahrscheinlichkeit

Daraus kann der notwendige Performance Level PLr abgeleitet werden.



Grafik: Der Risikograf nach ISO 13849

Legende:
(1) Risiko niedrig
(2) Risiko hoch
(3) Maßnahmen zur Risikominderung

Legende:

S = Wie schwer ist die mögliche Verletzung?
S1 = leichte (üblicherweise reversible) Verletzungen
S2 = schwere (üblicherweise nicht reversible) Verletzungen, einschließlich Tod

F = Wie oft und wie lange ist die Gefährdung? [1])
F1 = Selten bis weniger häufig und/oder die Dauer der Gefährdung ist kurz
F2 = Häufig bis dauernd und/oder die Dauer der Gefährdung ist lang

P = Kann die Gefährdung von der Person vermieden werden? [2])
P1 = realistisch möglich unter bestimmten Bedingungen
P2 = kaum möglich

PLr = erforderlicher Performance-Level
a = niedriger Beitrag zur Risikominderung
...
e = hoher Beitrag zur Risikominderung

[1]) Bei der Häufigkeit ist es nebensächlich, ob immer dieselbe Person gefährdet ist oder mehrere Personen nacheinander dem Risiko ausgesetzt sind.

[2]) Kann die Gefährdung von der betroffenen Person rechtzeitig erkannt oder vermieden werden oder kann die Auswirkung eines Unfalls deutlich gemindert werden? Das hängt u.a. von folgenden Aspekten ab:
- Betrieb mit oder ohne Beaufsichtigung,
- Betrieb durch Fachpersonal oder Laien,
- langsames oder schnelles Auftreten der Gefährdung,
- gute oder schlechte Möglichkeit, der Gefährdung durch Flucht zu entgehen,
- praktische Erfahrungen mit der Sicherheit eines solchen Prozesses.

Die Ermittlung des PLr wird daher einen gewissen Aufwand verursachen und die Einstufung in die einzelnen Zweige des Risikografen wird nicht immer eindeutig sein.

► Auch deshalb die Entscheidungswege der Risikobeurteilung klar dokumentieren und anschließend archivieren!

Stufen des Performance Level

13276

Die Fähigkeit, eine Sicherheitsfunktion unter vorhersehbaren Bedingungen auszuführen, wird einer der 5 Stufen des Performance-Level PL (PL a...PL e) zugeordnet. Dieser Performance-Level wird definiert als die Wahrscheinlichkeit eines gefahrbringenden Ausfalls der Sicherheitsfunktion (Sensor – Steuerungslogik – Aktor) je Stunde.

Performance-Level	Wahrscheinlichkeit eines gefahrbringenden Ausfalls [1/h]
PL a	$\geq 10^{-5}$ ... $< 10^{-4}$
PL b	$\geq 3 \cdot 10^{-6}$ ... $< 10^{-5}$
PL c	$\geq 10^{-6}$ ... $< 3 \cdot 10^{-6}$
PL d	$\geq 10^{-7}$ ... $< 10^{-6}$
PL e	$\geq 10^{-8}$ ... $< 10^{-7}$

Der **ecomat*mobile***-SafetyController ist für den Einsatz in Anwendungen mit Sicherheitsfunktionen (Sensor – Steuerungslogik – Aktor) bis zum Performance Level d geeignet.

3.2.2 Die vorgesehenen Architekturen der Maschinenfunktionen

13282

Zusammenfassung

► Kategorie der Maschinenfunktion wählen!
Dementsprechend die Sensoren, Aktuatoren und Ein-/Ausgänge der Sicherheitssteuerung verschalten und das Anwendungsprogramm für die Sicherheitsfunktion programmieren!
Dabei beachten, wie sich die Sicherheitsfunktion auch unter Fehlerbedingungen verhält!

- Cat.2-Strukturen:
Sensoren und Aktoren sind meist einkanalig angeschlossen.
In regelmäßigen Zeitabständen – 100 mal häufiger, als die Sicherheitsfunktion im Betrieb benötigt wird – testen, ob die Sicherheitsfunktion noch fehlerfrei arbeitet. (z.B. Schalter betätigen, Diagnosefunktion der Steuerung nutzen)!
- Cat.3-Strukturen:
Meist sind jeweils 2 Sensoren und 2 Aktuatoren an jeweils 2 Ein- und 2 Ausgängen angeschlossen. Die Steuerung prüft in der Anwendung zyklisch die Gültigkeit der Signale.

► Zusätzlich zum vorgenannten Performance Level PL der einzelnen Komponenten (Sensor, Steuerung, Aktor) den strukturellen Aufbau (die Kategorie "Cat.") der Maschinenfunktion (Sicherheitsfunktion) beachten. Die Kategorie bestimmt das Verhalten der Sicherheitsfunktion bezüglich der Widerstandsfähigkeit gegen Fehler.

Dieser strukturelle Aufbau wird in 5 Stufen (= Kategorien, → folgende Tabelle) eingeteilt. Die Kategorien beschreiben den mechanischen, hydraulischen und elektrischen Aufbau der Sicherheitsfunktion.

Kategorie	Anforderung	Systemverhalten
Cat. B	Das System muss den zu erwartenden Einflüssen in der mobilen Maschine standhalten können. Grundlegende Sicherheitsprinzipien müssen verwendet werden.	Ein Fehler kann zum Verlust der Sicherheitsfunktion führen.
Cat. 1	Anforderung von Cat. B muss erfüllt sein. Bewährte Bauteile und Prinzipien bei der Umsetzung der Mechanik und dem elektrischen Aufbau verwenden.	Systemverhalten von Cat. B, aber mit geringerer Fehlerwahrscheinlichkeit.
Cat. 2	Die Anforderungen von Cat. B und die Verwendung bewährter Sicherheitsprinzipien müssen erfüllt sein. Die Sicherheitsfunktion muss in geeigneten Zeitabständen durch die Maschinensteuerung getestet werden.	Das Auftreten eines Fehlers kann zum Verlust der Sicherheitsfunktion zwischen den Tests führen. Der Verlust der Sicherheitsfunktion wird durch den Test erkannt
Cat. 3	Die Anforderungen von Cat. B und die Verwendung bewährter Sicherheitsprinzipien müssen erfüllt sein. Sicherheitsbezogene Teile müssen wie folgt gestaltet werden: • ein einzelner Fehler in jedem dieser Teile darf nicht zum Verlust der Sicherheitsfunktion führen • wenn immer in angemessener Weise durchführbar, muss der einzelne Fehler erkannt werden.	Wenn ein einzelner Fehler auftritt, bleibt die Sicherheitsfunktion immer erhalten. Einige, aber nicht alle Fehler werden erkannt. Eine Anhäufung von unerkannten Fehlern kann zum Verlust der Sicherheitsfunktion führen.

Kategorie	Anforderung	Systemverhalten
Cat. 4	Die Anforderungen von Cat. B und die Verwendung bewährter Sicherheitsprinzipien müssen erfüllt sein. Sicherheitsbezogene Teile müssen wie folgt gestaltet werden: • ein einzelner Fehler in jedem dieser Teile darf nicht zum Verlust der Sicherheitsfunktion führen • der einzelne Fehler muss bei oder vor der nächsten Anforderung der Sicherheitsfunktion erkannt werden. Wenn diese Erkennung nicht möglich ist, darf eine Anhäufung von unerkannten Fehlern nicht zum Verlust der Sicherheitsfunktion führen.	Wenn ein einzelner Fehler auftritt, bleibt die Sicherheitsfunktion immer erhalten. Die Erkennung von Fehleranhäufungen reduziert die Wahrscheinlichkeit des Verlustes der Sicherheitsfunktion (⇨ hohe DC). Die Fehler werden rechtzeitig erkannt, um einen Verlust der Sicherheitsfunktion zu verhindern.

3.2.3 Mit dem V-Modell das Erstellen der sicheren Maschine organisieren

13264

Zusammenfassung

► Arbeitsschritte im V-Modell definieren, einhalten, überprüfen und dokumentieren
► Verantwortlichkeiten für Aufgaben festlegen
► Verantwortliche Person für die Sicherheitstechnik benennen (Functional Safety Manager)

Jeder Maschinenhersteller sollte durch organisatorische Maßnahmen innerhalb seines Unternehmens die Verantwortung für die einzelnen Aufgaben regeln. Dies ist unabhängig von den reinen Arbeitsschritten ...

- bei der Konstruktion der Maschine
- beim Erstellen des Anwendungsprogramms für den SafetyController

Es sollte auch ein verantwortlicher Mitarbeiter für die Sicherheitstechnik (FSM = Functional Safety Manager) benannt werden. Bei größeren Unternehmen wird diese Funktion einem hauptverantwortlichen Mitarbeiter zugeordnet.

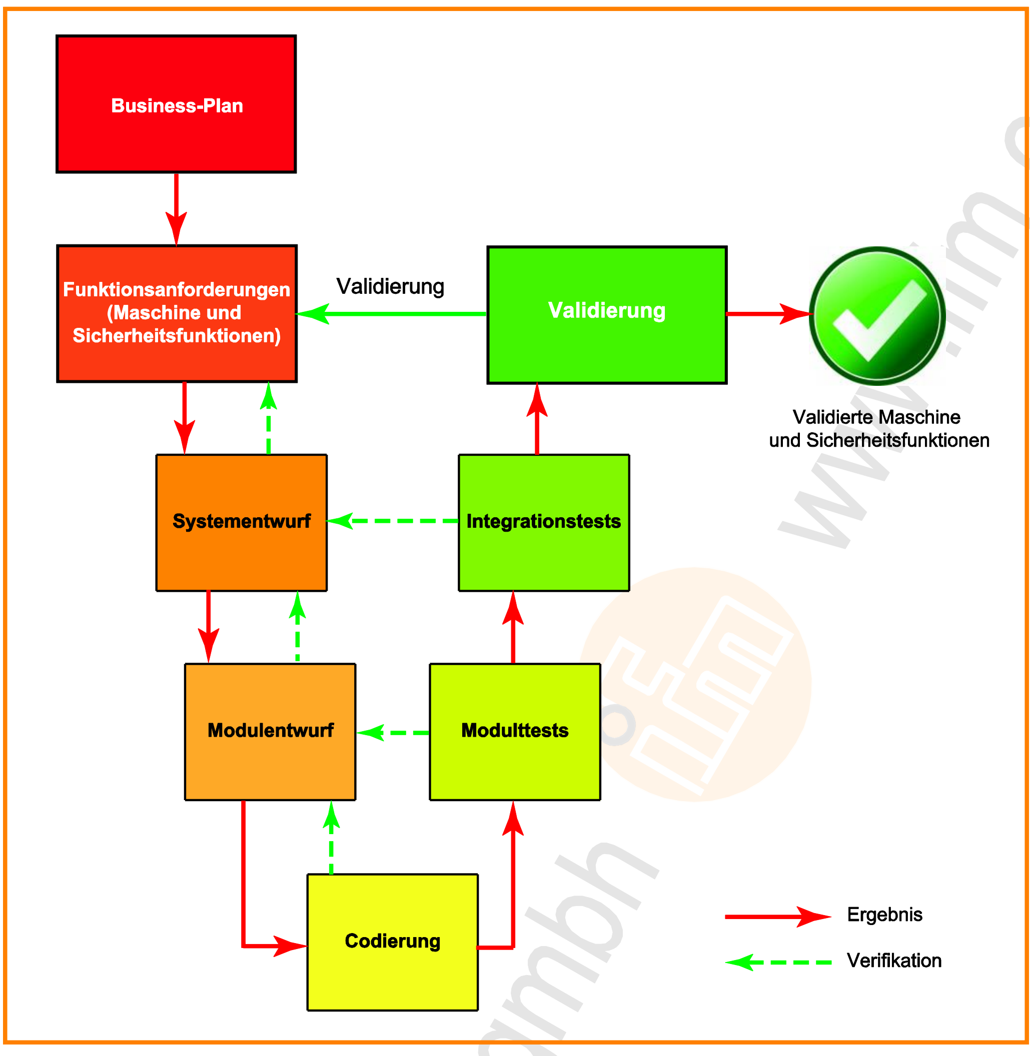
► Aufgaben:
  - Lastenheft erstellen
  - Sicherheitskonzept und Maschinenspezifikation erstellen
  - Funktionsspezifikation erstellen
  - Zuverlässigkeit der Sicherheitsfunktion ermitteln oder berechnen
  - die einzelnen Arbeitsschritte dokumentieren
  - Dokumentation archivieren und über die Lebensdauer der Maschine aufbewahren

► Das Durchführen einer Aufgabe und die Prüfung dieser Aufgabe darf nicht derselbe Mitarbeiter erledigen!
  - das Lastenheft erstellen und prüfen
  - die Spezifikation der Maschine erstellen und prüfen
  - die einzelnen sicherheitsrelevanten Funktionseinheiten erstellen und prüfen
  - das Zusammenspiel der Funktionseinheiten prüfen

  Hier arbeitet man typisch nach dem "4-Augen-Prinzip".

Um diese Struktur grafisch zu verdeutlichen, kann man das so genannte "V-Modell" anwenden. Die einzelnen Schritte muss der Maschinenhersteller für den konkreten Fall mit Details und den verantwortlichen Personen versehen. Der Maschinenhersteller könnte auch – je nach Arbeitspaketen – mehrere Organisationseinheiten nach dem V-Modell aufstellen.

Wichtig ist, dass diese Struktur der Organisation dokumentiert und archiviert wird.



Grafik: V-Modell mit den einzelnen Arbeitsschritten

- ► Entsprechend den derzeit gültigen Richtlinien und Normen:
  - bei der Konstruktion der mobilen Maschine die mechanische und elektrische Umsetzung der Sicherheitsfunktionen beachten
  - die einzelnen Schritte bei der Spezifikation (Lastenheft/Pflichtenheft) und ihrer Umsetzung nachvollziehbar dokumentieren
  - eine Aussage zu deren Zuverlässigkeit und zu einem möglichen Auftreten eines gefährlichen Fehlers machen
- ► Diese Unterlagen über die gesamte Lebensdauer der mobilen Arbeitsmaschinen oder der Maschinenserie archivieren!

3.2.4 Unterstützung und Prüfung durch externe Organisationen

13263

Zusammenfassung

- ► Bei Projektbeginn festlegen, ob externe Unterstützung und Zertifizierung benötigt wird.
- ► Verantwortlichkeiten für das Maschinen- und Sicherheitskonzept festlegen.

Der Hersteller einer mobilen Arbeitsmaschine muss entscheiden, ob das sicherheitstechnische Konzept für diese Maschine durch eine externe Organisation bewertet und zusätzlich geprüft werden soll.
Für einige Anwendungen ist die Prüfung und Zertifizierung durch eine unabhängige Prüfstelle gesetzlich vorgeschrieben.

Bei den meisten mobilen Arbeitsmaschinen liegt die Verantwortung beim Hersteller. Er selber kann entscheiden, ...

- ob er die Maschinenentwicklung komplett selber verantwortet
- oder ob er seine Arbeit überprüfen lässt.

Unabhängig davon, welcher Weg eingeschlagen wird:
Beim Maschinenbauer liegt die Gesamtverantwortung für ...

- das Maschinenkonzept,
- die Identifikation der Sicherheitsfunktionen und das Sicherheitskonzept
- die damit verbundenen Kennwerte (PL und Kategorie),
- die Wahl der Komponenten und
- das Überprüfen (Validieren) der Sicherheitsfunktionen.

Dieser gesamte Prozess muss nachvollziehbar dokumentiert und archiviert werden.

Die externe Organisation wird diese Arbeit bewerten und gegebenenfalls korrigierend eingreifen. Sie stellt in den meisten Fällen darüber auch ein Zertifikat aus.

Eine Entwicklung ohne externe Unterstützung ist genau so aufwendig wie mit Unterstützung, denn die Arbeitsschritte sind die gleichen. Jedoch kann die Entwicklung ohne externe Unterstützung im Schadenfall gegebenenfalls zu erheblichen rechtlichen Konsequenzen für den Maschinenbauer führen, da der Nachweis über den ordnungsgemäßen Projektablauf nicht oder nur schwer geführt werden kann.

3.3 Sicherheitstechnologie beim SafetyController

Inhalt

13383

Hier stellen wir Ihnen den **ecomat*mobile***-SafetyController für den Einsatz in mobilen Arbeitsmaschinen vor:

- die Hardware-Struktur des SafetyControllers
- wie werden die Signale verarbeitet?
- wie werden Fehler erkannt?
- wie reagiert das Gerät auf Fehler?
- was müssen Konstrukteur und Anwendungsprogrammierer zur Sicherheit beitragen?
- welche **ifm**-Software-Bausteine sind für sichere Anwendungen zertifiziert?

3.3.1 Sicherheitsarchitektur

Inhalt

13386

Realisierte Sicherheitsarchitektur

13387

Der Hardwareaufbau der Steuerung entspricht einer Realisierung nach IEC 62061 / IEC 61508 mit einer Hardware-Fehler-Toleranz (= HFT) von "0" (⇨ HFT = 0).

Das Gerät erreicht folgende Sicherheitskennwerte:

- Performance Level d oder SIL CL 2 für eine 2-kanalige Anschaltung der sicherheitsrelevanten Ein- und Ausgänge
- SIL CL 1 bei einer 1-kanaligen Anschaltung der sicherheitsrelevanten Ein- und Ausgänge

Realisiert ist dies durch eine Gerätearchitektur mit folgenden Strukturmerkmalen:

- modifizierte 1oo1-HW-Architektur (1 out of 1 architecture) mit separater Testeinrichtung
- zentrale Logikeinheit/Verarbeitungseinheit, mit der Möglichkeit, Signale zu bearbeiten
- wählbar:
  - 2-kanalige sichere Ein-/Ausgänge
  - optional 1-kanalige sichere Ein-/Ausgänge
- Testeinrichtung zur Überwachung der Haupt-CPU und der Spannungsversorgungen mit zentralem Abschaltausgang

Aufgrund dieser Basis-Architektur nimmt die Steuerung im Fehlerfall standardmäßig den "sicheren Zustand" ein. Je nach Abhängigkeit des Fehlers kann der Anwender eine andere, schwächere Fehlerreaktion mittels Keep-Alive konfigurieren (→ Kapitel ***Keep-Alive-Funktionalität*** (→ Seite 74)).

ℹ Der sichere Zustand eines Ausgangs ist der energielose Zustand (Low-Signal, "0"). Im Fehlerfall stellt das Gerät an seinen Ausgängen keine Energie mehr zur Verfügung ("de-energize-to-trip").

HINWEIS

Bei Einsatz des SafetyControllers muss die Summe von Diagnose-Testintervall und der Zeit, die zum Erreichen oder Aufrechterhalten eines sicheren Zustands benötigt wird, kleiner sein als die durch die spezifische Anwendung ("sichere Maschine") erforderliche ***Prozesssicherheitszeit*** (→ Seite 481, "***Die Prozesssicherheitszeit***" → Seite 35)!

Somit kann der ecomatmobile-SafetyController bei 2-kanaliger Realisierung der I/O-Anschaltung und entsprechender Programmierung mit der Steuerung Anwendungen der Kategorie 3 realisieren.

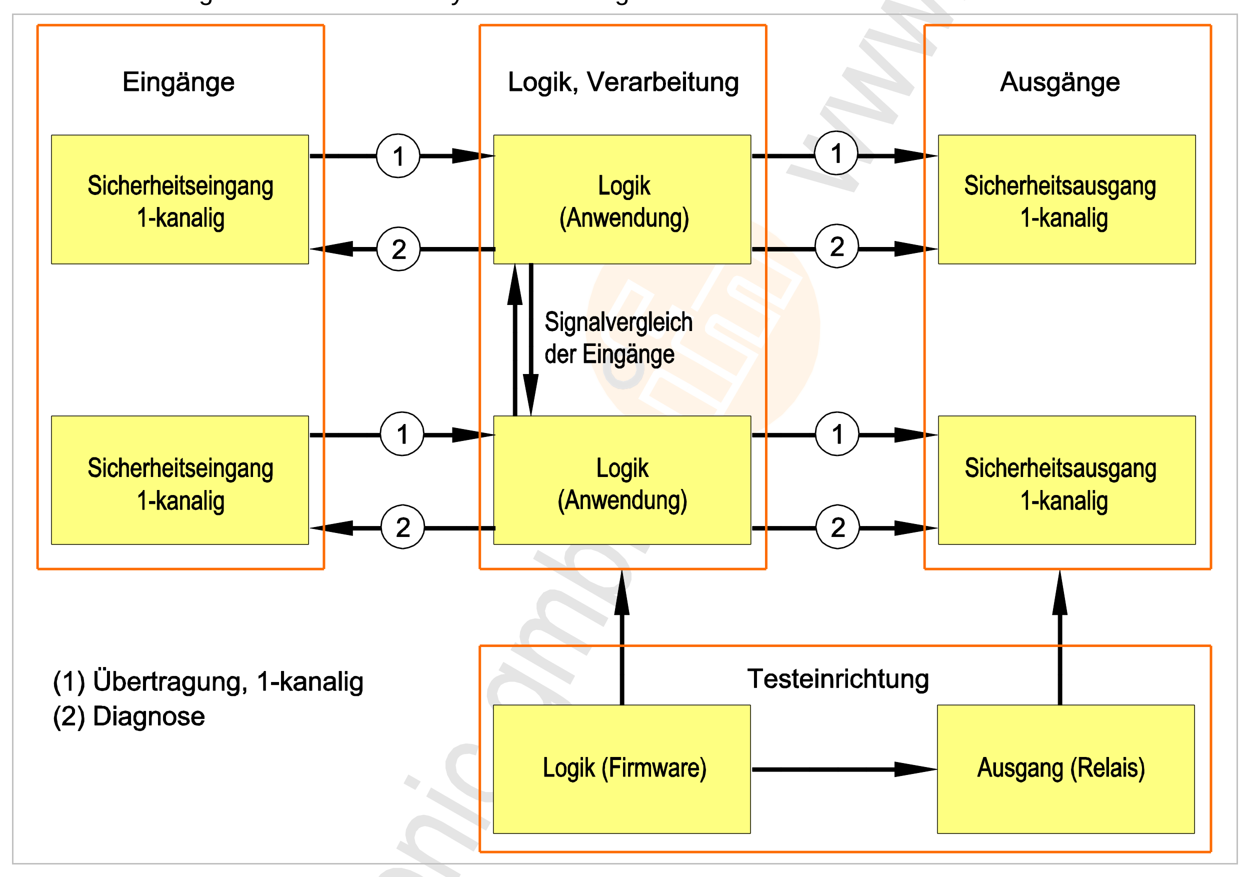
Sicherheitsarchitektur nach EN 13849-1

13388

Zur Verwendung des SafetyController in einer Anwendung basierend auf einer Kategorie der EN 13849-1 wird der Systemaufbau mit der Beschreibung des Systemverhaltens gemäß den Kategorien verglichen.

► Auf Grund der Flexibilität des SafetyControllers und der durch den Anwender bereitgestellten Anwendung:
  - den physikalischen Aufbau betrachten
  - die Möglichkeiten der Anwendungsprogrammierung berücksichtigen!

Eine Anwendung kann mit dem SafetyController folgenden strukturellen Aufbau realisieren:



13455

Das Gerät ist so gestaltet, dass jeweils ein Eingang über die Verarbeitungslogik im Anwendungsprogramm auf einen Ausgang geführt werden kann. Das ermöglicht den Aufbau einer 2-kanalige Programmstruktur. Die zentralen Teile der Steuerung (CPU, Speicher, Spannungsversorgung) werden gemeinsam genutzt.

Das Anwendungsprogramm muss in diesem Fall die Signale der Eingänge miteinander vergleichen, und so feststellen, ob ein Fehler an den miteinander korrespondierenden Eingängen vorliegt.

Entsprechendes gilt für die Ausgänge.

Erfolgen innerhalb des Anwendungsprogramms weitere Berechnungen, so sollten auch diese auf 2 Wegen durchgeführt und anschließend verglichen werden.

Wird bei einem Vergleich ein Fehler erkannt, muss im Anwendungsprogramm der sichere Zustand (durch Abschalten beider Ausgänge) eingenommen werden.

Die Prozesssicherheitszeit

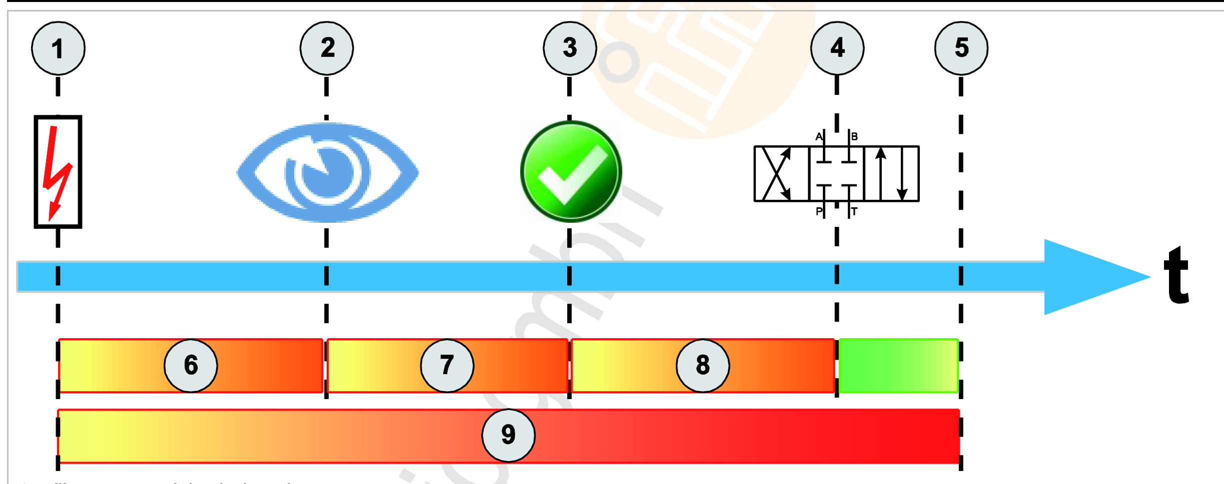
13284

Zusammenfassung

- ► Zulässige Prozesssicherheitszeit der Sicherheitsfunktion ermitteln.
- ► Prüfen, ob die Anwendung im Fehlerfall innerhalb der Prozesssicherheitszeit den sicheren Zustand erreichen kann.
- Die Sicherheitszeit des SafetyControllers muss kürzer sein als die Prozesssicherheitszeit der Sicherheitsfunktion.

► Bei der Auslegung der Sicherheitsfunktion zusätzlich auf die Prozesssicherheitszeit der Anwendung achten!
Nach einem Fehler in einer Sicherheitsfunktion muss die Anwendung innerhalb der Prozesssicherheitszeit reagiert und den sicheren Zustand erreicht haben.

Die zulässige Prozesssicherheitszeit muss länger sein als die Summe aus...
- Fehlererkennungszeit
- Fehlerreaktionszeit im SafetyController
- Aktuator-Schaltzeit



Grafik: Prozesssicherheitszeit

Legende:
(1) Fehlereintritt in der Sicherheitsfunktion
(2) SafetyController hat Fehler erkannt
(3) Sicherheitsstatus im SafetyController erreicht
(4) Prozess im sicheren Zustand
(5) Prozesssicherheitszeit abgelaufen
(6) Fehlererkennungszeit
(7) Fehlerreaktionszeit im SafetyController
(6)+(7) Sicherheitszeit des SafetyControllers
(8) Schaltzeiten der Aktuatoren
(9) zulässige Prozesssicherheitszeit

- ► Mögliche weitere Verzögerungen durch vor- bzw. nachgeschaltete Komponenten (Sensoren, Aktuatoren) bei der zeitlichen Auslegung berücksichtigen. Diese Zeiten verlängern die Fehlerreaktionszeit.
  Werden Daten der Sicherheitsfunktion über das Netzwerk von einem SafetyController zu einem anderen übertragen, dann die sich hierdurch verlängerte Fehlererkennungs- und Fehlerreaktionszeit berücksichtigen!
- ► Für die anderen Komponenten in der Sicherheitsfunktion → Datenblätter der Hersteller.

Bei einfachen Komponenten (z.B. Schalter, Ventile) gilt:

- ► Die Diagnosefunktion der Steuerung für Ein- und Ausgänge nutzen!

> $\text{Sicherheitszeit}_{\text{gesamt}} = \text{Sicherheitszeit}_{\text{SafetyController}} + \text{Schaltzeiten}_{\text{Aktuatoren}}$
> z.B. für Ventile: Schaltzeit ca. 200 ms

Sicherheitszeit = Fehlererkennungszeit + Fehlerreaktionszeit

Für den **ecomat*mobile***-SafetyController gilt:
$\text{Sicherheitszeit}_{\text{SafetyController}} \leq 210$ ms (bei Zykluszeit $\leq$ 66 ms)
$\text{Sicherheitszeit}_{\text{SafetyController}} \leq 310$ ms (bei Zykluszeit > 66 ms).

Ausnahmen:
• sichere Frequenzeingänge: → Kapitel ***Frequenzeingänge prüfen*** (→ Seite 60)

> Für sowohl ein- als auch zweikanalige Verwendung gilt:
> ► Die im Anwendungsprogramm implementierten Diagnosemaßnahmen in jedem IEC-Zyklus ausführen, damit die $\text{Sicherheitszeit}_{\text{SafetyController}}$ erreicht werden kann!

Wenn, wie oben beschrieben, die Sicherheitszeit kleiner ist als die durch die Sicherheitsfunktion der mobilen Arbeitsmaschine geforderte Prozesssicherheitszeit, kann ein einzelner Fehler (im ungünstigen Fall) kurzzeitig zu einem fehlerhaften Ausgangssignal führen, aber nicht zum Verlust der Sicherheitsfunktion.

Zum Verlust der Sicherheitsfunktion kann es nur kommen, wenn das fehlerhafte Signal nicht innerhalb der Prozesssicherheitszeit korrigiert werden kann.

- ► Durch die Gestaltung der spezifischen Anwendung im Anwendungsprogramm das Einhalten der Prozesssicherheitszeit sicherstellen!

Ein einzelner Fehler in der Steuerung kann in folgenden Fällen nicht zu einer Gefährdungssituation führen:

- wenn der sichere Zustand (Ausgänge AUS) eingenommen wird...
  • nach einem erkannten Fehler oder
  • bei der nächsten Anforderung der Sicherheitsfunktion oder
  • vor der nächsten Anforderung der Sicherheitsfunktion.
- wenn das Erkennen des Fehlers und die Reaktion auf den Fehler innerhalb der Prozesssicherheitszeit erfolgen.

Der Diagnosedeckungsgrad DC und der MTTFd-Wert

13283

Zusammenfassung

MTTFd ist eine wichtige Kenngröße zur Wahrscheinlichkeit eines gefährlichen Ausfalls einer Komponente.

DC gibt an, wie gut eine Komponente in der Lage ist, einen gefährlichen Ausfall rechtzeitig zu erkennen.

► MTTFd der einzelnen Komponenten aus dem jeweiligen Datenblatt des Herstellers ermitteln. Bei einfachen Komponenten → Tabellen der ISO 13849.

► DC der einzelnen Komponenten aus dem jeweiligen Datenblatt des Herstellers ermitteln.

Die Wahrscheinlichkeit eines Ausfalls der Sicherheitsfunktion, bei der es zu einer Gefährdung kommt, hängt neben der Architektur auch von der Zuverlässigkeit der verwendeten Bauteile ab.

Die Zuverlässigkeit wird bestimmt durch folgende Faktoren:

- die Zuverlässigkeit der verwendeten Bauteile
  wird dargestellt als durchschnittliche Zeit bis zum gefahrbringenden Ausfall "MTTFd" (= Mean Time To Failure, dangerous), → Datenblatt
  für die Sensoren und Aktoren: → Dokumentation des Herstellers
  (falls keine Werte vorhanden → Tabellen der EN 13849)
- der Umfang der Fehlererkennungs-Mechanismen, der Diagnose-Deckungsgrad "DC"

Da es sich bei dem ***ecomatmobile***-SafetyController um eine zertifizierte Mobilsteuerung handelt, wurde schon bei der Entwicklung auf die richtige Spezifikation und Überprüfung des Gerätes und der Gerätesoftware geachtet. Das wird mit der Bauartprüfung bescheinigt. So ist der Maschinenhersteller entlastet und kann sich auf die Erstellung der reinen Sicherheitsfunktionen im Anwendungsprogramm und die richtige Integration des SafetyControllers in die Maschine konzentrieren.

► Der SafetyController ist frei programmierbar. Der Anwendungsprogrammierer ist verantwortlich für die gemäß der Sicherheitsanforderung in der Anwendung erforderlichen Diagnoseaufgaben, z.B.:
  • Vergleich von zwei Eingängen
  • am Eingang oder Ausgang erlaubte Wertebereiche

- Der Diagnosedeckungsgrad (DC) des gesamten SafetyControllers ist "mittel" (DC = 90...99 %).
- Die MTTFd ist "mittel" (10 Jahre < MTTFd < 30 Jahre).

Anhäufung unentdeckter Fehler

13458

Aufgrund des Aufbaus des SafetyControllers kann davon ausgegangen werden, dass die Anhäufung unentdeckter Fehler zu einem Verlust der Sicherheitsfunktion führen kann. Das Systemverhalten des SafetyControllers ist daher wie folgt:

- Bei Erkennen eines einzelnen Fehlers wird die Sicherheitsfunktion immer ausgeführt oder der sichere Zustand wird eingenommen.
- Einige, aber nicht alle Fehler werden erkannt.
- Die Anhäufung unerkannter Fehler kann zum Verlust der Sicherheitsfunktion führen (und damit zu einer Gefährdungssituation).

Dieses Gerät eignet sich für den Einsatz in sicherheitsrelevanten Anwendungen:
• bis zu PL d nach ISO 13849,
• bis zu SIL CL 2 nach IEC 62061.

! Voraussetzungen:
• 2-kanalige Realisierung der Ein-/Ausgangs-Anschaltung
• entsprechende Programmierung der Steuerung

3.3.2 Betriebszustände / Betriebsarten des Controllers

Inhalt

13296

Betriebszustände

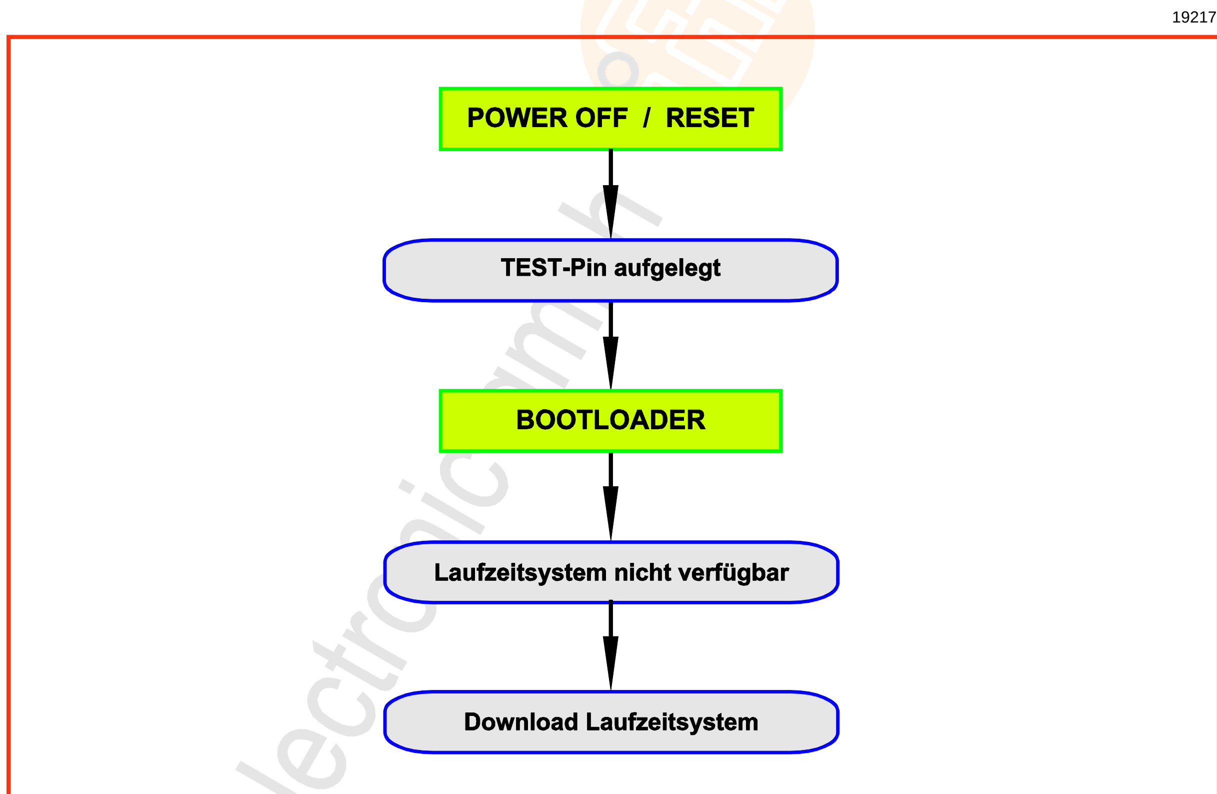
1075

Nach Anlegen der Versorgungsspannung kann sich das **ecomat*mobile***-Gerät in einem von fünf möglichen Betriebszuständen befinden:

- BOOTLOADER
- INIT
- STOP
- RUN
- SYSTEM STOP

Betriebszustände

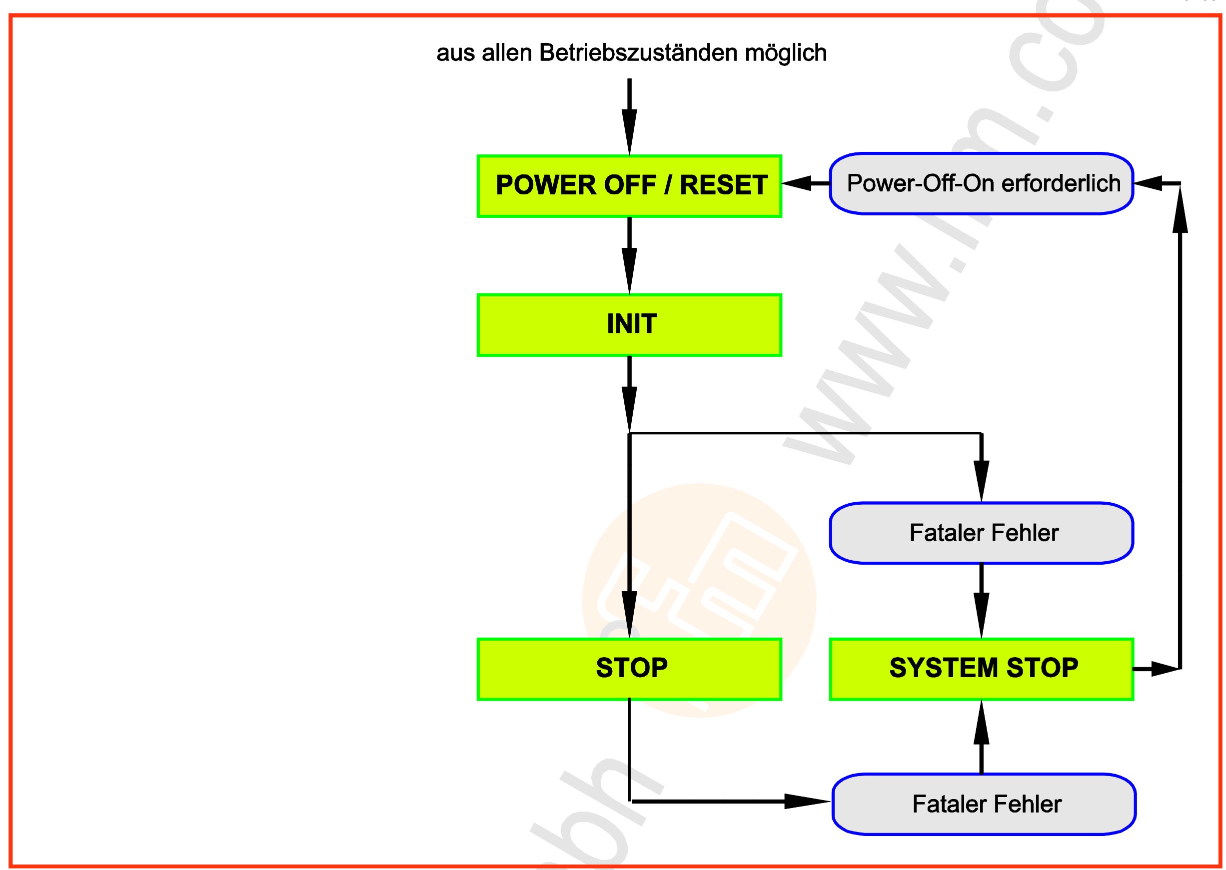
19217



Grafik: Betriebszustände (hier: Laufzeitsystem ist nicht verfügbar)

Betriebszustände: Anwendungsprogramm nicht verfügbar

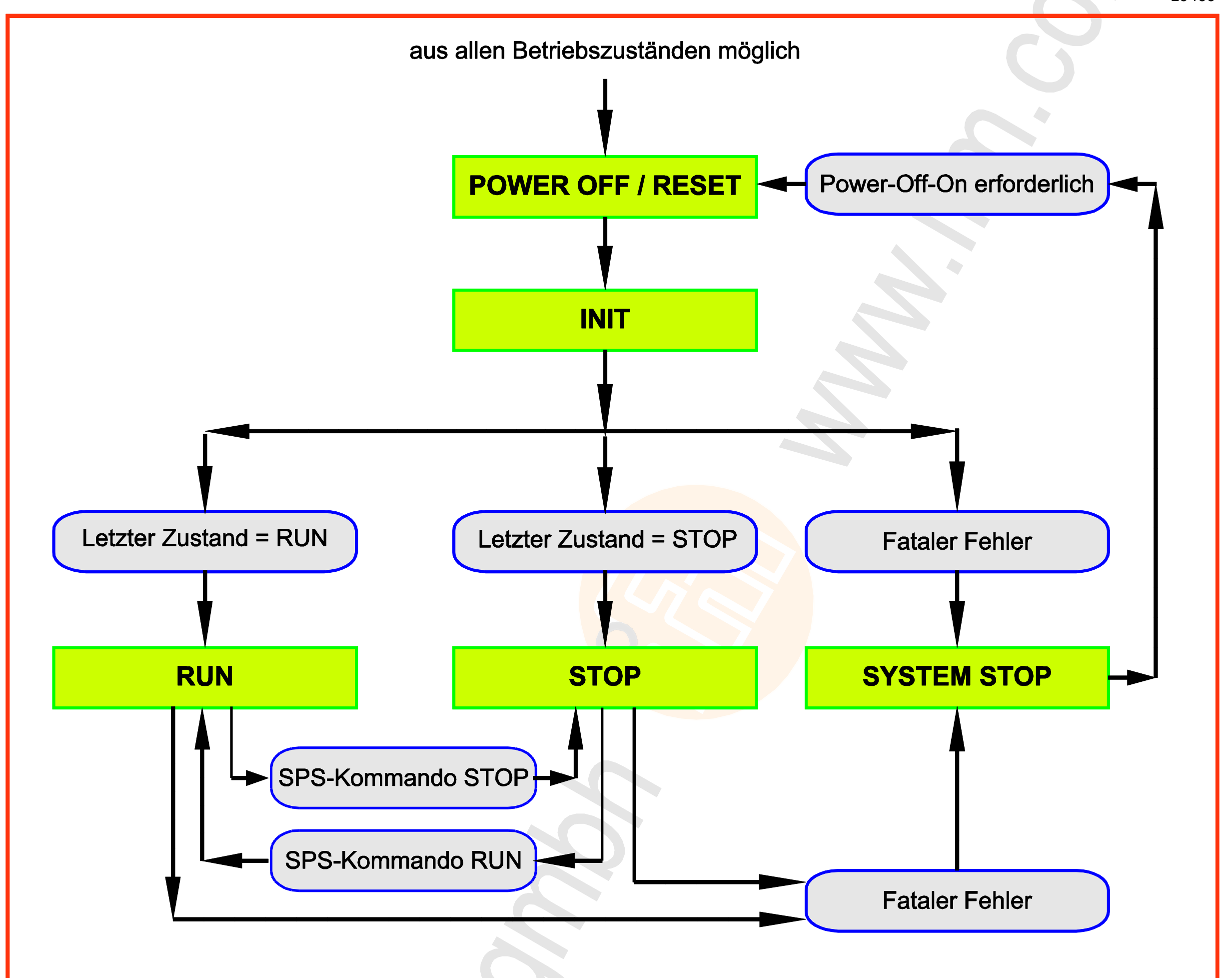
19465



Grafik: Betriebszustände (hier: Anwendungsprogramm ist nicht verfügbar)

Betriebszustände: Anwendungsprogramm verfügbar

19466



Grafik: Betriebszustände (hier: Anwendungsprogramm ist verfügbar)

Bootloader-Zustand

1080

Es wurde kein Laufzeitsystem geladen. Der **ecomat*mobile***-Controller befindet sich im Bootloader-Zustand. Vor dem Laden des Anwendungsprogramms muss ein Laufzeitsystem-Download durchgeführt werden.

> Die LED blinkt grün (5 Hz).

INIT-Zustand (Reset)

1076

Voraussetzung: ein gültiges Laufzeitsystem ist installiert.

Dieser Zustand wird nach jedem Power-On-Reset durchlaufen:

> Das Laufzeitsystem wird initialisiert.

> Verschiedene Checks werden durchgeführt, z.B. Warten auf gültige Versorgungsspannung.

> Dieser nur temporäre Zustand wird vom RUN- oder STOP-Zustand abgelöst.

> Die LED leuchtet orange.

Wechsel aus diesem Zustand in einen der folgenden Zustände möglich:
• RUN
• STOP

STOP-Zustand

1078

Dieser Zustand wird in folgenden Fällen erreicht:

- Aus dem RESET-Zustand, wenn:
  • kein Anwendungsprogramm ist geladen oder
  • der letzte Zustand vor dem RESET-Zustand war der STOP-Zustand
- Aus dem RUN-Zustand durch das STOP-Kommando
  • nur bei Betriebsmodus = TEST (→ Kapitel ***TEST-Betrieb*** (→ Seite 43))

> Die LED leuchtet grün.

RUN-Zustand

1077

Dieser Zustand wird in folgenden Fällen erreicht:

- Aus dem RESET-Zustand, wenn:
  • der letzte Zustand vor dem RESET-Zustand war der RUN-Zustand
- Aus dem STOP-Zustand durch das RUN-Kommando
  • nur bei Betriebsmodus = TEST (→ Kapitel ***TEST-Betrieb*** (→ Seite 43))

> Die LED blinkt grün (2 Hz).

SYSTEM-STOP-Zustand

10447

In diesen Zustand fällt der **ecomat*mobile***-Controller, wenn ein nicht tolerierbarer Fehler (FATAL ERROR) festgestellt wurde. Dieser Zustand kann nur durch einen Power-Off-On-Reset verlassen werden.

> Die LED erlischt.

Betriebsmodi

1083

Unabhängig von den Betriebszuständen kann der Controller in verschiedenen Betriebsmodi betrieben werden.

TEST-Betrieb

13703

HINWEIS

► Erst NACH dem Anschließen des OPC-Client den TEST-Anschluss mit der Versorgungsspannung verbinden!

> Ansonsten tritt ein fataler Fehler auf.

Dieser Betriebsmodus wird durch Anlegen von Versorgungsspannung am Test-Eingang erreicht (→ Montageanleitung > Kapitel "Technische Daten" > Kapitel "Anschlussbelegung").

Jetzt kann der Controller im RUN- oder STOP-Zustand Kommandos über eine der Schnittstellen entgegennehmen und z.B. mit dem Programmiersystem kommunizieren.

Zusammenfassung Test-Eingang ist aktiv:
- Programmiermodus ist freigeben
- Software-Download ist möglich
- die sicheren Ausgänge sind deaktiviert
- es werden keine CANsafety-Nachrichten versendet
- Zustand des Anwendungsprogramms ist abfragbar
- kein Schutz der gespeicherten Software möglich

► Zum Wiederaktivieren der sicherheitsrelevanten Ausgänge und der CANsafety-Telegramme:
1. Test-Eingang = LOW
2. PowerOn-Reset durchführen

Monitoring- oder Debug-Modus

13705

Controller-Eingang TEST	FB SET_DEBUG Eingang DEBUG	Variablenwerte	als "SAFETY" konfigurierte Ausgänge
nicht mit VBB verbunden	TRUE	MONITORING-Modus: nur lesen möglich	aktiv
mit VBB verbunden	TRUE oder FALSE	DEBUG-Modus: lesen und ändern möglich	deaktiviert

Tabelle: Verhalten der sicherheitsrelevanten Ausgänge im MONITORING- und DEBUG-Modus

HINWEIS

Der Debug-Modus kann bestehen bleiben, auch wenn das Anwendungsprogramm ohne erneutes Verwenden von SET_DEBUG aktualisiert wurde.
> Ein fortgesetzter Lesezugriff kann möglich sein.
> Ein fortgesetzter Schreibzugriff ist nicht mehr möglich.
► Nach Aktualisieren des Anwendungsprogramms ein Power-On-Reset durchführen!
Somit wird der Debug-Modus zuverlässig unterbrochen.

Details:

SET_DEBUG (→ Seite 378)	organisiert (abhängig vom TEST-Eingang) den DEBUG-Modus oder den Monitoring-Modus

Test-Eingang → Montageanleitung > Kapitel "Technische Daten" > Kapitel "Anschlussbelegung"

3.3.3 Überwachungs- und Sicherungsmechanismen

Inhalt

3926

Für diese Geräte laufen automatisch folgende Überwachungen ab:

Nach Einschalten der Versorgungsspannung

13928

Nach dem Einschalten der Versorgungsspannung (Steuerung ist im Bootloader) laufen im Gerät folgende Tests ab:

> RAM-Test (einmalig)
> Versorgungsspannung
> Systemdaten-Konsistenz
> CRC des Bootloaders
> wenn vorhanden und gestartet: CRC des Laufzeitsystems
> wenn vorhanden und gestartet: CRC des Anwendungsprogramms

Programmablauf und Zykluszeit überwachen

13399

Der SafetyController hat eine interne Programmablauf-Überwachung. Diese Überwachung generiert (unter anderem) daraus das interne Watchdog-Signal (→ ***Watchdog*** (→ Seite 487)).

> Eine Testeinrichtung überwacht dieses Signal auf eine Periodendauer ≤ 100 ms.
> Bei Erkennen eines Ausfalles setzt die Testeinrichtung die Verarbeitung der Logik (CPU) in den sicheren Zustand (⇨ fataler Fehler).
  ℹ Sicherer Zustand = die Testeinrichtung deaktiviert die CPU und die Relais.
  ⇨ alle Ausgänge sind stromlos.

ℹ Die Laufzeitüberwachung erfolgt über die komplette Betriebssoftware, bestehend aus...
• Laufzeitsystem und
• Anwendungsprogramm.

Überwachung vor dem Zugriff

13929

Folgende Parameter überwacht das Gerät vor dem Zugriff auf die Daten:

- die CANsafety-Konfiguration

Zyklische Überwachung

13930

Folgende Parameter überwacht das Gerät zyklisch:

- Gerätetemperatur
  Im Fehlerfall:
  • ERROR_TEMPERATURE = TRUE
- Versorgungsspannung
  Im Fehlerfall:
  • ERROR_POWER = TRUE oder
  • ERROR_VBBr = TRUE oder
  • ERROR_VBBo = TRUE
- CAN-Busse
- Systemdaten:
  • Laufzeitsystem geladen und unverändert
  • Anwendungsprogramm geladen und unverändert
  • Systemtakt
  • Node-ID
  • Baudrate von CAN und RS232
- Speichertests:
  • RAM
  • Flash
- Relaisfunktion:
  • Überwachung erfolgt, solange das Relais geöffnet sein soll.
- sichere Eingänge:
  • Querschluss zu anderen sicheren Eingängen
  • "stuck at 1" / "stuck at 0" (= dauerhaftes TRUE / FALSE)
  • Überstrom (nur bei Stromeingängen)
- sichere Ausgänge:
  • Querschluss zu anderen sicheren Ausgängen
  • Schluss gegen Versorgung
  • "stuck at 1"
  • Kurzschluss
  • Leiterbruch
  • Überlast

Kontinuierliche Überwachung

13534

Folgende Parameter überwacht das Gerät kontinuierlich:

- interne Versorgungsspannungen

Wenn TEST-Pin nicht aktiv

13391

> Schreibschutz für Systemdaten im FRAM [1]), z.B.:
  • Laufzeitsystem geladen,
  • Kalibrierdaten.
  Realisiert über Hard- und Software.
> Schreibschutz für Anwendungsprogramm (im Flash-Speicher)

[1]) FRAM steht hier allgemein für alle Arten von nichtflüchtigen, schnellen Speichern.

Einmalige Mechanismen

3930

> CRC-Überwachung bei Download oder Upload.
> Überprüfung der Gerätezugehörigkeit von Laufzeitsystem und Anwendungsprogramm.

3.3.4 Fehler erkennen und verarbeiten

Inhalt

13406

Fehlerklassen

Inhalt

13407

Abhängig von den möglichen Auswirkungen eines Fehlers wird dieser in eine Klasse eingeordnet. Abhängig von der Fehlerklasse ergibt sich ein entsprechendes Verhalten des Systems auf den Fehler, wenn er auftritt.

Hier unterscheiden wir:

- allgemeine Fehler
- schwere Fehler
- fatale Fehler

Allgemeine Fehler

13408

Als allgemeine Fehler gelten folgende Fehler:
• bei ihrem Auftreten ist kein direkter Handlungsbedarf des Systems erforderlich,
• die Reaktion auf den Fehler wird der Anwendung überlassen.

Reaktion auf allgemeine Fehler:

- das Auftreten des Fehlers wird der Anwendung über Merker und/oder Fehler-Code signalisiert,
- die Anwendung kann den Fehler zurücksetzen.

Schwere Fehler

13409

Als schwere Fehler gelten folgende Fehler:
• beim Auftreten des Fehlers werden Sicherheitsaspekte berührt
• der Fehler erlaubt aber die weitere Ausführung der Anwendung.

Reaktion auf schwere Fehler:

> das Auftreten des Fehlers wird der Anwendung über Merker und/oder Fehler-Code signalisiert

> wird ein schwerer Fehler erkannt:
• die Steuerung geht in den sicheren Zustand.
Sicherer Zustand = Das Laufzeitsystem deaktiviert alle sicheren Ausgänge und stoppt das Senden von CANsafety-Nachrichten.
Ausnahmen:
a) sichere Ausgänge und CANsafety-Schnittstellen, die mit ***SET_KEEP_ALIVE*** (→ Seite 386) ausgenommen wurden
b) CANsafety-Schnittstellen senden weiter, wenn schwere Fehler ausschließlich an sicheren Ausgängen vorliegen

> der Fehler wird über die Geräte-LED signalisiert, wenn mindestens ein sicherer Ausgang oder ein CANsafety-Kanal in den sicheren Zustand gebracht wurde

- die Anwendung kann den Fehler zurücksetzen.

Fatale Fehler

13410

Als fatale Fehler gelten folgende Fehler:

- beim Auftreten des Fehlers werden Sicherheitsaspekte berührt
- der Fehler ist so gravierend, dass die weitere Ausführung der System-Software oder der Anwendung nicht mehr zulässig ist.

Reaktion auf fatale Fehler:

> weder das Laufzeitsystem noch die Anwendung werden weiter ausgeführt

> der Fehler veranlasst den sicheren Zustand.
Sicherer Zustand = das Laufzeitsystem stoppt die Steuerung (fataler Fehler):
• alle Ausgänge werden abgeschaltet
• die Abarbeitung der Software wird angehalten
• es ist keine Kommunikation mehr möglich
• die LED erlischt.

► Für Neustart des Geräts:
• Fehlerursache beseitigen
• Power-On-Reset durchführen.

Wenn der TEST-Eingang bereits zuvor aktiv war: • alle sicherheitsrelevanten Ausgänge sind deaktiviert • alle ausgehenden CANsafety-Nachrichten sind deaktiviert • fatale Fehler wirken nur wie schwere Fehler Wenn im Fehlerfall noch möglich: • die Anwendung läuft weiter • Fehlerdiagnose ist möglich • Kommunikation ist möglich

Fehlermeldung

13411

Fehlermeldung der Anwendung

13842

In der Anwendung erkannte Fehler können als Fehler-Code an das Laufzeitsystem gemeldet werden, das entsprechend der gemeldeten Fehlerklasse reagiert. Die Übermittlung der anwendungsspezifischen Fehler erfolgt über den FB ***ERROR_REPORT*** (→ Seite 382).

Für den Anwender vereinfacht sich hierdurch die Reaktion auf einen Fehler, da er sich in der Anwendung nicht selber um das Deaktivieren der Ausgänge kümmern muss, sondern dem System die Aufgabe überträgt, inkl. der Handhabung von Keep-Alive (→ Kapitel ***Keep-Alive-Funktionalität*** (→ Seite 74)).

Die aktuell aktiven Fehlercodes können wie folgt ausgelesen werden:
- über den Systemmerker ERRORCODE
- über den FB ***SHOW_ERROR_LIST*** (→ Seite 389)

Fehlermeldung durch das Laufzeitsystem

13843

Fehler, die durch das Laufzeitsystem selbst erkannt werden, werden intern behandelt. Die dazugehörenden Fehler-Codes werden bei einigen IEC-Funktionsbausteinen über einen Ausgang der Anwendung übergeben.

Die aktuell aktiven Fehlercodes können wie folgt ausgelesen werden:
- über den Systemmerker ERRORCODE
- über den FB ***SHOW_ERROR_LIST*** (→ Seite 389)

Sicherer Zustand

13413

Was ist ein "sicherer Zustand"? Der Zustand einer Maschine gilt als sicher, wenn von ihr keine →Gefährdung mehr ausgeht. Dies ist meist der Fall, wenn alle gefahrbringenden Bewegungsmöglichkeiten abgeschaltet sind und nicht unerwartet wieder anlaufen können.

In diesem Zusammenhang unterscheiden wir für den SafetyController folgende Vorgänge:

Die Anwendung deaktiviert Ausgänge

13414

Sicherer Zustand = Das Anwendungsprogramm deaktiviert alle entsprechend programmierten Ausgänge.
Falls ein anwendungsspezifischer Fehler mit dem FB ***ERROR_REPORT*** (→ Seite 382) an das Laufzeitsystem gemeldet wurde:
Das Laufzeitsystem deaktiviert alle sicheren Ausgänge und stoppt das Senden von CANsafety-Nachrichten. **Nicht** deaktiviert werden die für diesen Fehler mit ***SET_KEEP_ALIVE*** (→ Seite 386) ausgenommenen Ausgänge und CANsafety-Schnittstellen.

Diese Reaktion betrifft Fehler, die nur durch die Anwendung erkannt werden können, z.B.:

- Vergleich der zweikanaligen Eingangssignale, denn nur die Anwendung kennt die Zuordnung der logischen Kanäle auf die physikalischen Eingänge.
- Prüfen des Wertebereiches von Anwendungssignalen.

► Eine mögliche Signalisierung des sicheren Zustands durch die Geräte-LED liegt in der Verantwortung des Programmierers.

Das Laufzeitsystem deaktiviert Ausgänge

13415

Sicherer Zustand = Das Laufzeitsystem deaktiviert alle sicheren Ausgänge und stoppt das Senden von CANsafety-Nachrichten.
Nicht deaktiviert werden die für diesen Fehler mit ***SET_KEEP_ALIVE*** (→ Seite 386) ausgenommenen Ausgänge und CANsafety-Schnittstellen.
Ausnahme: Melden nur sicherheitsrelevante Ausgänge einen schweren Fehler, dann werden CANsafety-Nachrichten weiter gesendet.

Da diese Fehler nicht zur kompletten Abschaltung des Systems führen, werden sie durch Ausfallmeldungen (Fehler-Codes) der Anwendung mitgeteilt.

Auswirkungen:

> Die LED blinkt rot mit der zuletzt konfigurierten Blinkfrequenz (voreingestellt: 2 Hz)

> (optional) mindestens ein sicherer Ausgang ist abgeschaltet

> (optional) das Senden von CANsafety-Nachrichten über mindestens eine Schnittstelle ist eingestellt.

Das Relais deaktiviert Ausgänge

13416

Der sichere Zustand ist in folgenden Fällen gewährleistet:
- das Relais der jeweiligen Ausgangsgruppe (VBBx) ist ausgeschaltet
- die Ausgänge werden somit nicht mehr mit Energie versorgt.

Ansonsten gelten dieselben Anforderungen wie bei "Laufzeitsystem deaktiviert Ausgänge".

Dieser sichere Zustand wird genutzt, wenn ein oder mehrere sichere Ausgänge einer Relaisgruppe den Fehler "stuck at 1" (= dauerhaftes TRUE) haben.

14179

HINWEIS

Damit das System das Relais im geöffneten Zustand überprüft, muss mindestens einmal innerhalb von 24 Stunden Folgendes geschehen:
- der SafetyController startet neu oder
- die Relais werden für länger als 100 ms geöffnet.

Sicherer Zustand: Übersicht

13424

Fehler / Fehlerklasse:	Anwendung erkennt Fehler	Schwerer Fehler	Sicherer Ausgang lässt sich nicht abschalten	Fataler Fehler	Unterspannung VBBs
Ausführende Komponente:	**Anwendung**	**Laufzeitsystem**	**Laufzeitsystem**	**System**	**Hardware**
Ausgeführte Aktionen des sicheren Zustandes:					
Sichere Ausgänge sind energielos	X[1]	X	X	X	X
Relais der Ausgangsgruppe ist energielos			X	X	X
Alle Ausgänge sind energielos			X	X	X
Senden von CANsafety-Telegrammen ist gestoppt	X[1]	X[3]	X[3]	X	X
Keep-Alive ist möglich	X[2]	X	X		
Die Ausführung der Anwendung wird gestoppt				X	X
Zurücksetzen des sicheren Zustandes ist nur durch PowerOn-Reset des Systems möglich				X	X[4]
Wiederinbetriebnahme einer deaktivierten Komponente nach dem Wegfall des Fehlers ist möglich	X	X	X		

[1]) Das Deaktivieren der sicheren Ausgänge und das Stoppen des Versands von CANsafety-Telegrammen in der Anwendung programmieren!
Alternativ über den FB ***ERROR_REPORT*** (→ Seite 382) dem Laufzeitsystem einen Anwendungsfehler melden, wodurch das Laufzeitsystem das Verhalten "schwerer Fehler" vornimmt.

[2]) Wird nicht über den FB ERROR_REPORT dem Laufzeitsystem ein Fehler gemeldet: in der Anwendung selbst einen vergleichbaren Mechanismus zu Keep-Alive implementieren.
Für schwere Fehler, die durch das Laufzeitsystem erkannt werden, kann aber nur der Keep-Alive-Mechanismus des Laufzeitsystems genutzt werden.

[3]) Das Senden von CANsafety-Telegrammen wird nicht gestoppt, wenn der sichere Zustand aufgrund eines oder mehrerer schweren Fehler an den sicheren Ausgängen eingenommen wurde.

[4]) Die Steuerung wird wieder in Betrieb genommen, sobald die Versorgungsspannung wieder hoch genug ist.

Fehler zurücksetzen

13412

Ein Fehler muss durch die Anwendung zurückgesetzt werden. Bis dahin verhält sich das System so, als wenn der Fehler noch aktiv sei, auch wenn der Fehler selbst nicht mehr vorhanden ist.

► Die aktiven Fehler ermitteln mit dem FB ***SHOW_ERROR_LIST*** (→ Seite 389).

► Den Fehler zurücksetzen mit dem FB ***ERROR_RESET*** (→ Seite 384).

Die betroffene Komponente wird nach dem Rücksetzen des Fehlers wieder in Betrieb genommen, da zu diesem Zeitpunkt nicht bestimmt werden kann, ob der Fehler noch vorliegt. Sollte der Fehler doch noch vorliegen, wird er nach Ablauf der entsprechenden Diagnosezeit wieder gemeldet.

Falls die betroffene Komponente zu ihrem Schutz zeitweise (⇨ Schutzzeit) deaktiviert wurde:
(Ursachen z. B.:
• Überstrom am Eingang,
• Überlast oder
• Kurzschluss am Ausgang)
dann wird der Fehler gleich im nächsten SPS-Zyklus wieder gemeldet.
Dann hat das Rücksetzen dieser Fehler während der Schutzzeit keine Wirkung.

Die beim Zurücksetzen des Fehlers angegebene Fehlerklasse wird nicht ausgewertet. Gründe:

- es kann keine aktiven Fehler-Codes geben, die sich nur in der Fehlerklasse unterscheiden,
- um das Rücksetzen von Fehlergruppen zu vereinfachen.

⚠ WARNUNG

Gefahr durch unbeabsichtigtes und gefährliches Anlaufen von Maschinen- oder Anlagenteilen!

► Der Programmierer muss bei der Programmerstellung verhindern, dass nach Auftreten eines Fehlers (z.B. NOT-HALT) und der anschließenden Fehlerbeseitigung unbeabsichtigt Maschinen- oder Anlagenteile gefährlich anlaufen können!
⇨ Wiederanlaufsperre realisieren!

► Dazu im Fehlerfall die in Frage kommenden Ausgänge im Programm logisch abschalten!

3.3.5 Sicherheitsrelevante Signale verarbeiten

Inhalt

13391

► Beim Aufbau des Anwendungsprogramms die sicherheitsrelevanten Signale von Ein- und Ausgängen, Funktionsbausteinen und der Hardware verarbeiten und überwachen!
► Dazu möglichst die vorhandenen Sicherheitsfunktionen des Laufzeitsystems und / oder der Safety-Bibliothek verwenden!

Eingänge für Sicherheitsfunktionen

Inhalt

13935

Im SafetyController sind nur folgende Eingänge für Sicherheitsfunktionen zugelassen (→ Datenblatt):

Gerät	Adressen der Sicherheitseingänge	Anzahl der Sicherheitseingänge
SafetyController: CR7032, CR7132	I00...I15	16

Beschreibung der Konfiguration → Kapitel ***Funktionskonfiguration der Ein- und Ausgänge*** (→ Seite 177).

Analogeingänge prüfen

13393

Die Analogeingänge können wie folgt verwendet werden:
• einkanalig (SIL CL 1) oder
• zweikanalig (SIL CL 2).

► Im FB ***SET_INPUT_MODE*** (→ Seite 264):
• für jeden sicherheitsrelevanten Eingang den Parameter SAFETY = TRUE setzen!

HINWEIS

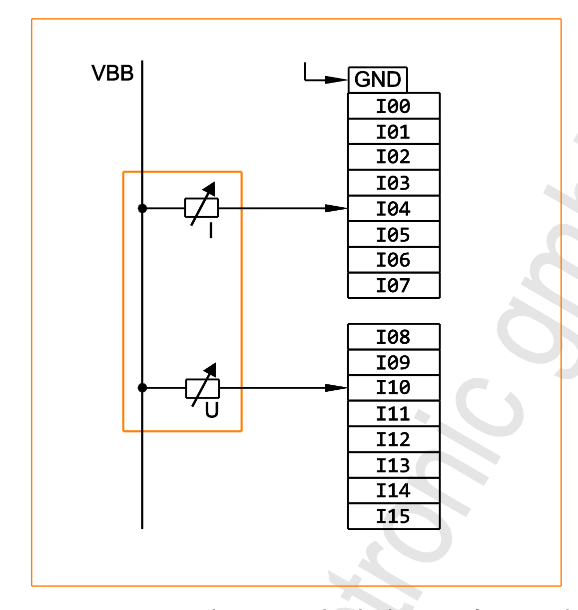
Nutzsignal-Frequenz an den sicheren Eingängen = maximal 34 Hz

► Sichere Eingänge nur mit Software-Filterstufe 4 betreiben!
→ Kapitel ***Software-Filter der Eingänge konfigurieren*** (→ Seite 179)

> Andernfalls setzt das Laufzeitsystem das Gerät in den sicheren Zustand.

Sicherer Zustand = Das Laufzeitsystem deaktiviert alle sicheren Ausgänge und stoppt das Senden von CANsafety-Nachrichten.
Nicht deaktiviert werden die für diesen Fehler mit ***SET_KEEP_ALIVE*** (→ Seite 386) ausgenommenen Ausgänge und CANsafety-Schnittstellen.

Bei 2-kanaligem Betrieb können auch Querschlüsse erkannt werden.



Beispiel analoger Sicherheitssensor:
hier: 2-kanalig und diversitär

Eingang I04 mit Stromsignal,
Eingang I10 mit Spannungssignal
desselben Sensors.

► Der Anwender muss für jeden analogen Eingang auch den zulässigen Wertebereich definieren. Dabei auch die Toleranzen der Signalgeber und Eingänge berücksichtigen!

► Im Anwendungsprogramm das Einhalten des gültigen Wertebereichs zyklisch überprüfen!

► Ferner ist es sinnvoll, die Signalspannung nur in einem begrenzten Bereich (z.B. 10...90 %) auszuwerten. Dadurch können folgende Fehler erkannt werden:
• Schluss gegen Masse (< 10 %)
• Leiterbruch (< 10 %)
• Schluss gegen Versorgungsspannung (> 90 %)

Zum Vergleichen der 2-kanalig ermittelten Signale geeignete Funktionsbausteine:

SF_EQUIVALENT_WORD (→ Seite 287)	• vergleicht zwei sichere Eingangswerte [WORD] miteinander • prüft die Werte auf zulässigen Wertebereich und zulässige Abweichung

► Bei Überschreiten einer zulässigen Abweichung zwischen den Kanälen (2-kanaliger Betrieb) oder des zulässigen Wertebereichs:
alle Sicherheitsfunktionen, die diese Signale verarbeiten, in den sicheren Zustand schalten!
ℹ Sicherer Zustand = Das Anwendungsprogramm deaktiviert alle entsprechend programmierten Ausgänge.
Falls ein anwendungsspezifischer Fehler mit dem FB ***ERROR_REPORT*** (→ Seite 382) an das Laufzeitsystem gemeldet wurde:
Das Laufzeitsystem deaktiviert alle sicheren Ausgänge und stoppt das Senden von CANsafety-Nachrichten. **Nicht** deaktiviert werden die für diesen Fehler mit ***SET_KEEP_ALIVE*** (→ Seite 386) ausgenommenen Ausgänge und CANsafety-Schnittstellen.

Binäreingänge prüfen

13394

Die Binäreingänge können wie folgt verwendet werden:
• einkanalig (SIL CL 1) oder
• zweikanalig (SIL CL 2).

► Im FB ***SET_INPUT_MODE*** (→ Seite 264):
  • für jeden sicherheitsrelevanten Eingang den Parameter SAFETY = TRUE setzen!

HINWEIS

Nutzsignal-Frequenz an den sicheren Eingängen = maximal 34 Hz

► Sichere Eingänge nur mit Software-Filterstufe 4 betreiben!
→ Kapitel ***Software-Filter der Eingänge konfigurieren*** (→ Seite 179)

> Andernfalls setzt das Laufzeitsystem das Gerät in den sicheren Zustand.

Sicherer Zustand = Das Laufzeitsystem deaktiviert alle sicheren Ausgänge und stoppt das Senden von CANsafety-Nachrichten.
Nicht deaktiviert werden die für diesen Fehler mit ***SET_KEEP_ALIVE*** (→ Seite 386) ausgenommenen Ausgänge und CANsafety-Schnittstellen.

Bei 2-kanaligem Betrieb können auch Querschlüsse erkannt werden.

► Der Anwender muss für einen binären Eingang den zulässigen zeitlichen Abstand beim Zustandswechsel (TRUE ⇨ FALSE / FALSE ⇨ TRUE) definieren.

► Im Anwendungsprogramm das Einhalten der Zeit beim Zustandswechsel zyklisch überprüfen!

Dabei helfen z.B. folgende FBs:

SF_ANTIVALENT (→ Seite 272)	• vergleicht zwei binäre sichere Eingänge miteinander • prüft den zeitlichen Ablauf der Eingänge zueinander
SF_EQUIVALENT (→ Seite 283)	• vergleicht zwei binäre sichere Eingänge miteinander • prüft den zeitlichen Ablauf der Eingänge zueinander

► Dauert der Zustandswechsel auf den Kanälen zu lange:
alle Sicherheitsfunktionen, die diese Signale verarbeiten, in den sicheren Zustand schalten!
Sicherer Zustand = Das Anwendungsprogramm deaktiviert alle entsprechend programmierten Ausgänge.
Falls ein anwendungsspezifischer Fehler mit dem FB ***ERROR_REPORT*** (→ Seite 382) an das Laufzeitsystem gemeldet wurde:
Das Laufzeitsystem deaktiviert alle sicheren Ausgänge und stoppt das Senden von CANsafety-Nachrichten. **Nicht** deaktiviert werden die für diesen Fehler mit ***SET_KEEP_ALIVE*** (→ Seite 386) ausgenommenen Ausgänge und CANsafety-Schnittstellen.

HINWEIS

Die Spannungsschwellen für die binären Signale liegen typischerweise bei...
• LOW = 30 % der Versorgungsspannung
• HIGH = 70 % der Versorgungsspannung

Bedingt durch die Toleranzen bei der Erfassung der Spannung am Eingang sollten bei sicherheitsrelevanten binären Eingängen die Signalgeber folgende Pegel einhalten:
• LOW = < 25 % der Versorgungsspannung
• HIGH = > 75 % der Versorgungsspannung

Binäreingänge für Sensoren nach NAMUR prüfen

8211

Diagnosefähige Sensoren nach NAMUR können ausgewertet werden.

► Die gleichen Überwachungen vornehmen wie bei 'Binäreingänge prüfen' angegeben!
Darüber hinaus:

► Im FB ***SET_INPUT_MODE*** (→ Seite 264):
• für jeden sicherheitsrelevanten Eingang den Parameter SAFETY = TRUE setzen!
• den Parameter DIAGNOSTICS = TRUE setzen!

HINWEIS

Nutzsignal-Frequenz an den sicheren Eingängen = maximal 34 Hz

► Sichere Eingänge nur mit Software-Filterstufe 4 betreiben!
→ Kapitel ***Software-Filter der Eingänge konfigurieren*** (→ Seite 179)

> Andernfalls setzt das Laufzeitsystem das Gerät in den sicheren Zustand.

Sicherer Zustand = Das Laufzeitsystem deaktiviert alle sicheren Ausgänge und stoppt das Senden von CANsafety-Nachrichten.
Nicht deaktiviert werden die für diesen Fehler mit ***SET_KEEP_ALIVE*** (→ Seite 386) ausgenommenen Ausgänge und CANsafety-Schnittstellen.

Für jeden sicherheitsrelevanten Binäreingang kann durch Setzen eines Konfigurationsbits zusätzlich die Diagnose aktiviert werden:

> Der SafetyController prüft zyklisch, dass der analoge Eingangsspannungswert nicht länger als 66 ms außerhalb des zulässigen Wertebereichs liegt.
zulässig: > 1 V...< 95 % von VBBs

> Die Diagnose erkennt Leitungsbrüche und Kurzschlüsse bei diagnosefähigen Sensoren nach NAMUR.

> Wird ein Fehler erkannt:
• ein schwerer Fehler wird gemeldet
• die Steuerung geht in den sicheren Zustand:
Sicherer Zustand = Das Laufzeitsystem deaktiviert alle sicheren Ausgänge und stoppt das Senden von CANsafety-Nachrichten.
Nicht deaktiviert werden die für diesen Fehler mit ***SET_KEEP_ALIVE*** (→ Seite 386) ausgenommenen Ausgänge und CANsafety-Schnittstellen.

Frequenzeingänge prüfen

10453

- ► Sicherheitsrelevante Frequenzeingänge nur 2-kanalig verwenden!
- ► Im FB ***SET_INPUT_MODE*** (→ Seite 264):
  • den Parameter SAFETY = FALSE setzen!
  In diesem Fall darf die Querschlusserkennung des Safety-Modus nicht aktiviert sein!

Geeignete Funktionsbausteine sind z.B.:

FREQUENCY (→ Seite 300)	misst die Frequenz des am gewählten Kanal ankommenden Signals
FREQUENCY_PERIOD (→ Seite 302)	misst die Frequenz und die Periodendauer (Zykluszeit) in [µs] am angegebenen Kanal
PERIOD (→ Seite 306)	misst am angegebenen Kanal die Frequenz und die Periodendauer (Zykluszeit) in [µs]
PERIOD_RATIO (→ Seite 308)	misst die Frequenz und die Periodendauer (Zykluszeit) in [µs] über die angegebenen Perioden am angegebenen Kanal. Zusätzlich wird das Puls-/Periodenverhältnis in [‰] angegeben.

- ► Der Anwender muss für jeden Frequenzeingang den zulässigen Wertebereich und die maximal zulässige Abweichung zwischen den Kanälen definieren.
  Dabei auch die Toleranzen der Signalgeber und Eingänge berücksichtigen!
- ► Im Anwendungsprogramm das Einhalten des gültigen Wertebereichs zyklisch überprüfen!

Zum Vergleichen der 2-kanalig ermittelten Signale geeignete Funktionsbausteine:

SF_EQUIVALENT_REAL (→ Seite 285)	• vergleicht zwei sichere Eingangswerte [REAL] miteinander • prüft die Werte auf zulässigen Wertebereich und zulässige Abweichung

- ► Bei Überschreiten einer vorzugebenden maximalen Abweichung zwischen den Kanälen oder des zulässigen Wertebereichs:
  alle Sicherheitsfunktionen, die diese Signale verarbeiten, in den sicheren Zustand schalten!
  Sicherer Zustand = Das Anwendungsprogramm deaktiviert alle entsprechend programmierten Ausgänge.
  Falls ein anwendungsspezifischer Fehler mit dem FB ***ERROR_REPORT*** (→ Seite 382) an das Laufzeitsystem gemeldet wurde:
  Das Laufzeitsystem deaktiviert alle sicheren Ausgänge und stoppt das Senden von CANsafety-Nachrichten. **Nicht** deaktiviert werden die für diesen Fehler mit ***SET_KEEP_ALIVE*** (→ Seite 386) ausgenommenen Ausgänge und CANsafety-Schnittstellen.

HINWEIS

Gefahr von Falschmessungen!

Bei Fehlern in der internen Takterzeugung kann die erreichbare Sicherheitszeit bis zu 2 s betragen.

Eine Drift der CPU-Frequenz um ± 1 MHz (das entspricht 80 kHz Drift des CPU-Quarzes) wird innerhalb der Sicherheitszeit von 4 s erkannt.

- ► Für eine geringere Sicherheitszeit:
  Einen weiteren Frequenzeingang mit einer geeigneten Referenz-Frequenz belegen und vergleichend auswerten.

Signale vergleichen

13257

Wird vom Sicherheitskonzept ein 2-kanaliger Aufbau der Hard- und Software gefordert, erfolgt der Signalvergleich der beiden Kanäle im Anwendungsprogramm.

► Der Anwender muss für jeden sicherheitsrelevanten Eingang den zulässigen Wertebereich und die maximal zulässige Abweichung zwischen den Kanälen definieren.
Dabei auch die Toleranzen der Signalgeber und Eingänge berücksichtigen!

► Im Anwendungsprogramm im 2-kanaligen Betrieb beide Kanäle im Programmablauf zyklisch miteinander vergleichen!

Zum Vergleichen der 2-kanalig ermittelten Signale geeignete Funktionsbausteine:

SF_ANTIVALENT (→ Seite 272)	• vergleicht zwei binäre sichere Eingänge miteinander • prüft den zeitlichen Ablauf der Eingänge zueinander
SF_EQUIVALENT (→ Seite 283)	• vergleicht zwei binäre sichere Eingänge miteinander • prüft den zeitlichen Ablauf der Eingänge zueinander
SF_EQUIVALENT_REAL (→ Seite 285)	• vergleicht zwei sichere Eingangswerte [REAL] miteinander • prüft die Werte auf zulässigen Wertebereich und zulässige Abweichung
SF_EQUIVALENT_WORD (→ Seite 287)	• vergleicht zwei sichere Eingangswerte [WORD] miteinander • prüft die Werte auf zulässigen Wertebereich und zulässige Abweichung

► Bei Überschreiten einer zulässigen Abweichung:
alle Sicherheitsfunktionen, die diese Signale verarbeiten, in den sicheren Zustand schalten!
ℹ Sicherer Zustand = Das Anwendungsprogramm deaktiviert alle entsprechend programmierten Ausgänge.
Falls ein anwendungsspezifischer Fehler mit dem FB ***ERROR_REPORT*** (→ Seite 382) an das Laufzeitsystem gemeldet wurde:
Das Laufzeitsystem deaktiviert alle sicheren Ausgänge und stoppt das Senden von CANsafety-Nachrichten. **Nicht** deaktiviert werden die für diesen Fehler mit ***SET_KEEP_ALIVE*** (→ Seite 386) ausgenommenen Ausgänge und CANsafety-Schnittstellen.

Eingänge für induktive Sicherheitssensoren

13135

Mit dem SafetyController können bis zu 8 Sicherheitsketten angesteuert und verarbeitet werden.

Jede Sicherheitskette darf jeweils aus maximal 9 induktiven Sicherheitssensoren in Reihe bestehen (also bis zu 72 Sicherheitssensoren).

Eine zusätzliche Auswerteelektronik ist dazu nicht erforderlich.

> Der SafetyController...
  • erzeugt das notwendige Taktsignal,
  • verarbeitet die Sensorausgangssignale.

► Sicherheitssensoren mit ***SAFETY_SWITCH*** (→ Seite 268) überwachen!

HINWEIS

Der SafetyController unterstützt z.Z. nur die folgenden Gerätetypen:
- Bestell-Nr. GG505S, zylindrisch, M18, Typ GIGA
- Bestell-Nr. GI505S, zylindrisch, M30, Typ GIIA
- Bestell-Nr. GM504S, quaderförmig, Typ GIMC
- Bestell-Nr. GM505S, quaderförmig, Typ GIMC

Alle Geräte müssen mit 24 V DC versorgt werden. Eine Nutzung der Sicherheitssensoren in Bordnetzen von 12 V DC ist nicht möglich.

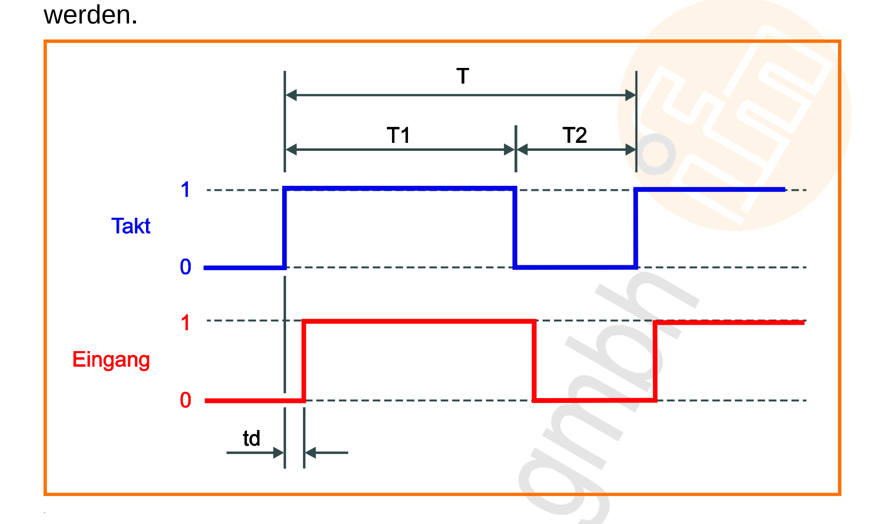
Funktionsweise

12254

> ❗ An den Eingängen, an die induktive Sicherheitssensoren angeschlossen werden, muss der Filter für Frequenzsignale deaktiviert sein (Ixx_DFILTER = 0).
> Ansonsten können nicht alle Fehler diagnostiziert werden.

Die Sicherheitssensoren werden mit einer Versorgungsspannung von 24 V DC versorgt. Zusätzlich müssen die Schalter ein Taktsignal von der Steuerung erhalten. Mittels des Taktsignals werden Verkabelungsfehler (Leiterbruch, Kurzschluss und Querschluss) und ein einfaches Umgehen des Schalters (z.B. durch Überbrücken von Taktsignal und Steuerungseingang) erkannt. Der Schalter überwacht und wertet das in der Steuerung erzeugte Taktsignal aus. Zusätzlich überwacht der Schalter die Versorgungsspannung und die ordnungsgemäße Position des Bedämpfungselementes.

Wird kein Fehler vom Schalter erkannt, wird das Taktsignal um ca. 1,5 ms (Zeit td) verzögert wieder als Eingangssignal dem SafetyController zur Verfügung gestellt. Der Zeitversatz und die korrekte Signalform werden von der Steuerung überwacht und ausgewertet. Bei Fehlerfreiheit wird der Ausgang der Softwarefunktion eingeschaltet und kann als digitales Eingangssignal weiterverarbeitet werden.



Typische Reaktionszeiten des SafetyControllers

12255

Angaben ohne Reaktionszeit des Sensors
Zykluszeit = max. 100 ms

T	= 255...325 ms + 2•Zykluszeit	= max. 525 ms
T1	= 200...270 ms + 1•Zykluszeit	= max. 370 ms
T2	= 55 ms + 1•Zykluszeit	= max. 155 ms
Td	= ca. 1,5 ms	
Reaktionszeit auf Sicherheitsanforderung (SWITCH_ON = FALSE)	= max. 55 ms + 1•Zykluszeit	= max. 155 ms
Reaktionszeit auf steigende Flanke des Sensorsignals (Sensor bedämpft)	= max. 325 ms + 2•Zykluszeit	= max. 525 ms
maximale Sicherheitszeit des SafetySwitch	= 2•(270 ms + 55 ms) + 1•Zykluszeit	= max. 750 ms

Weitere technische Daten → Gerätebeschreibung des jeweiligen Sensors.

Zulässige Ein- / Ausgänge

12256

Taktsignal: Zulässige Ausgänge

Gerät	zulässige Ausgangsadressen für Taktsignal	Anzahl zulässige Ausgänge
SafetyController: CR7032, CR7132	Q00...Q15	16

Sicherheitssensoren: zulässige Signaleingänge

Gerät	zulässige Eingangsadressen für Sicherheitssensoren	Anzahl zulässige Eingänge
SafetyController: CR7032, CR7132	I00...I07	8

Anzahl unabhängige Sicherheitsketten / Sicherheitssensoren:

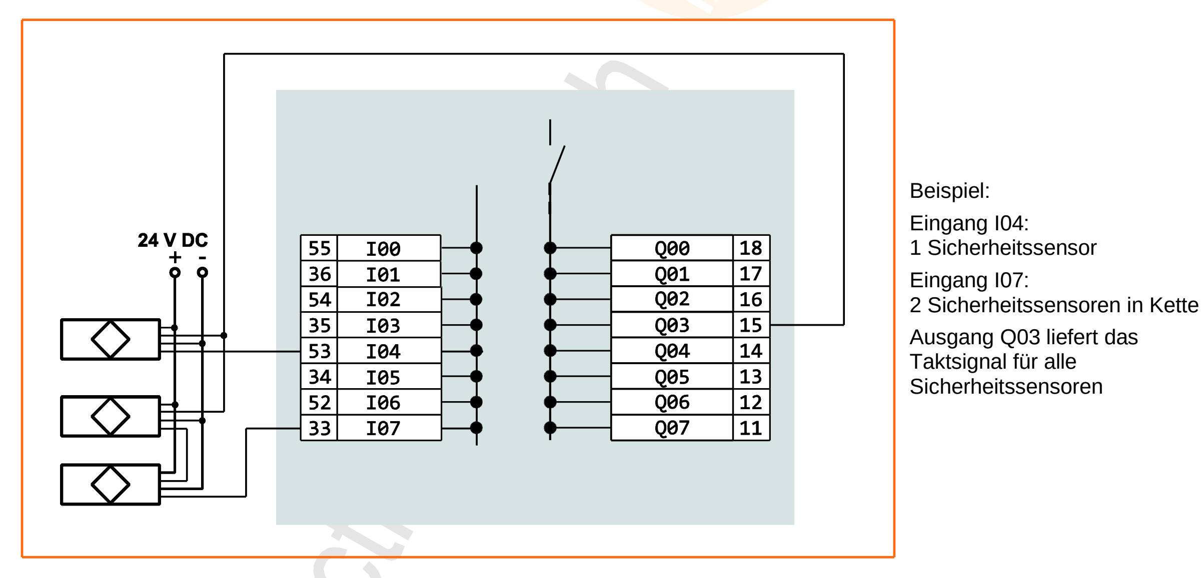
Gerät	max. Anzahl Sicherheitsketten	max. Anzahl Sicherheitssensoren je Sicherheitskette
SafetyController: CR7032, CR7132	8	9

max. Anzahl Sicherheitssensoren = (max. Anzahl Ketten) • (max. Anzahl Sensoren je Kette)

Beispiel: Sicherheitssensor

12264

Anschluss von 2 Sicherheitsketten mit insgesamt 3 induktiven Sicherheitssensoren:



► Sicherheitssensoren mit dem FB ***SAFETY_SWITCH*** (→ Seite 268) überwachen!

Ausgänge für Sicherheitsfunktionen

Inhalt

13936

Im SafetyController sind nur folgende Ausgänge für Sicherheitsfunktionen zugelassen (→ Datenblatt):

Gerät	Adressen der Sicherheitsausgänge	Anzahl der Sicherheitsausgänge
SafetyController: CR7032, CR7132	Q00...Q15	16

Beschreibung der Konfiguration → Kapitel ***Funktionskonfiguration der Ein- und Ausgänge*** (→ Seite 177).

> ! Zuerst die sicherheitsrelevanten Ausgänge konfigurieren!
> Erst danach dürfen die nicht-sicherheitsrelevanten Ausgänge konfiguriert werden.

Überlast erkennen

13396

► Im FB ***SET_OUTPUT_MODE*** (→ Seite 313): für jeden sicherheitsrelevanten Ausgang...
• den Parameter SAFETY = TRUE setzen!
• den Parameter DIAGNOSTICS = TRUE setzen!
Falls SAFETY = TRUE und DIAGNOSTICS = FALSE ⇨ Fehlermeldung!

Der Laststrom der Ausgänge wird über eine Strommessung überwacht.

> Der SafetyController deaktiviert den betroffenen Ausgang, wenn...
• der zulässige Laststrom wird für > 66 ms um mehr als 12,5 % überschritten.

HINWEIS

Bei Strommessung: die Filtereinstellungen Qxx_FILTER wirken sich auf die Diagnosezeit aus.

► Qxx_FILTER für die sicheren Ausgänge auf den voreingestellten Werten belassen!

> Eine erkannte Überlast signalisiert der SafetyController dem Anwendungsprogramm als Ausfallmeldung.

> Wird an einem sicheren Ausgang ein Fehler erkannt:
• ein schwerer Fehler wird gemeldet
• die Steuerung geht in den sicheren Zustand:
Sicherer Zustand = Das Laufzeitsystem deaktiviert alle sicheren Ausgänge. **Nicht** deaktiviert werden die für diesen Fehler mit ***SET_KEEP_ALIVE*** (→ Seite 386) ausgenommenen Ausgänge.
Bei schweren Fehlern ausschließlich an sicheren Ausgängen werden die CANsafety-Nachrichen weiterhin gesendet!

Fehler zurücksetzen und Ausgang reaktivieren:

> Für sichere Ausgänge gilt:
• der Fehler darf nicht mehr anliegen
• seit Fehlerbeginn muss ≥ 1 s vergangen sein
• der Fehler-Code muss danach zurückgesetzt worden sein
erst dann aktiviert der SafetyController den Ausgang wieder.

Kurzschluss erkennen

13398

► Im FB ***SET_OUTPUT_MODE*** (→ Seite 313): für jeden sicherheitsrelevanten Ausgang...
• den Parameter SAFETY = TRUE setzen!
• den Parameter DIAGNOSTICS = TRUE setzen!
Falls SAFETY = TRUE und DIAGNOSTICS = FALSE ⇨ Fehlermeldung!

> Ein Schluss gegen Masse kann nur erkannt werden bei Ausgang = TRUE.

Falls Versorgungsspannung < 7,3 V: an den Ausgängen keine Kurzschlusserkennung möglich!

Die festgelegte Diagnosezeit = 66 ms.

> Wird durch den Schluss gegen Masse der Ausgangstreiber thermisch überlastet, schaltet er sich zeitweise ab. Dieses Abschalten kann ebenfalls erfasst werden.

> Ein erkannter Kurzschluss wird dem Anwendungsprogramm per Ausfallmeldung signalisiert. In der Folge können weitere Fehlermeldungen (z.B. Überlast, Leiterbruch) erzeugt werden. Alle diese Meldungen erscheinen nacheinander in der Liste des FB ***SHOW_ERROR_LIST*** (→ Seite 389).

> Wird an einem sicheren Ausgang ein Fehler erkannt:
• ein schwerer Fehler wird gemeldet
• die Steuerung geht in den sicheren Zustand:
Sicherer Zustand = Das Laufzeitsystem deaktiviert alle sicheren Ausgänge. **Nicht** deaktiviert werden die für diesen Fehler mit ***SET_KEEP_ALIVE*** (→ Seite 386) ausgenommenen Ausgänge.
Bei schweren Fehlern ausschließlich an sicheren Ausgängen werden die CANsafety-Nachrichten weiterhin gesendet!

Fehler zurücksetzen und Ausgang deaktivieren:

> Für sichere Ausgänge gilt:
• der Fehler darf nicht mehr anliegen
• seit Fehlerbeginn muss ≥ 1 s vergangen sein
• der Fehler-Code muss danach zurückgesetzt worden sein
erst dann aktiviert der SafetyController den Ausgang wieder.

Querschluss erkennen

13973

► Im FB ***SET_OUTPUT_MODE*** (→ Seite 313): für jeden sicherheitsrelevanten Ausgang...
 • den Parameter SAFETY = TRUE setzen!
 • den Parameter DIAGNOSTICS = TRUE setzen!
 ⓘ Falls SAFETY = TRUE und DIAGNOSTICS = FALSE ⇨ Fehlermeldung!

► Je nach dem Ergebnis der Risikobeurteilung der Anwendung die sicheren Ausgänge auf folgende Fehler testen:
 • Querschluss untereinander und
 • Schluss gegen Versorgungsspannung.

> Dazu werden automatisch vom Laufzeitsystem der Steuerung nacheinander die überwachten (=rücklesbaren) Ausgänge mit einem kurzen Abschaltimpuls (ca. 250 µs) beaufschlagt. Dieser wird durch die integrierten Diagnosekanäle zurückgelesen und ausgewertet. Diese Testung erkennt innerhalb von 1 s, ob ein Querschluss zu einem anderen Kanal oder ein Schluss gegen Versorgung vorliegt.

> Durch diese Testung wird bei aktivem Ausgang der Querschluss in den sicheren Ausgängen erkannt.

> Ein durch die Testung erkannter Fehler wird über einen Fehlercode mit der Fehlerursache 0x0A (Safetydiagnose Ausgang) und dem entsprechenden Ausgang als Fehlerquelle ausgegeben.

> Wenn ein Fehler erkannt wird, wird das zum Ausgang gehörende Relais abgeschaltet (= zweiter Abschaltweg).

> Wird an einem sicheren Ausgang ein Fehler erkannt:
 • ein schwerer Fehler wird gemeldet
 • die Steuerung geht in den sicheren Zustand:
 ⓘ Sicherer Zustand = Das Laufzeitsystem deaktiviert alle sicheren Ausgänge. **Nicht** deaktiviert werden die für diesen Fehler mit ***SET_KEEP_ALIVE*** (→ Seite 386) ausgenommenen Ausgänge.
 ⓘ Bei schweren Fehlern ausschließlich an sicheren Ausgängen werden die CANsafety-Nachrichten weiterhin gesendet!

> Bei einem Schluss gegen Versorgungsspannung, der durch die Verkabelung verursacht wird, kommt es zu einem Folgefehler bedingt durch die Rückspeisung, die durch die externe Spannung am Ausgang verursacht wird. Die Relais-Überwachung prüft, ob das Relais den erwünschten Zustand hat. In dem Fall bei abgeschaltetem Relais würde die Relais-Überwachung noch eine Spannung hinter dem Relais sehen und dies als fatalen Fehler werten (→ Kapitel ***Fatale Fehler*** (→ Seite 50)). Damit wird die Steuerung komplett angehalten.

Fehler zurücksetzen und Ausgang deaktivieren:

► Für sichere Ausgänge gilt:
 • der Fehler darf nicht mehr anliegen
 • seit Fehlerbeginn muss ≥ 1 s vergangen sein
 • der Fehler-Code muss danach zurückgesetzt worden sein
 erst dann aktiviert der SafetyController den Ausgang wieder.

Leiterbruch erkennen

13397

► Im FB ***SET_OUTPUT_MODE*** (→ Seite 313): für jeden sicherheitsrelevanten Ausgang...
• den Parameter SAFETY = TRUE setzen!
• den Parameter DIAGNOSTICS = TRUE setzen!
Falls SAFETY = TRUE und DIAGNOSTICS = FALSE ⇨ Fehlermeldung!

Ein Strom von < 25 mA bei Ausgang = TRUE wird als Leiterbruch interpretiert.

Die festgelegte Diagnosezeit = 66 ms.

HINWEIS

Bei Strommessung: die Filtereinstellungen Qxx_FILTER wirken sich auf die Diagnosezeit aus.

► Qxx_FILTER für die sicheren Ausgänge auf den voreingestellten Werten belassen!

> Einen erkannten Leiterbruch signalisiert der SafetyController dem Anwendungsprogramm als Ausfallmeldung.

> Wird an einem sicheren Ausgang ein Fehler erkannt:
• ein schwerer Fehler wird gemeldet
• die Steuerung geht in den sicheren Zustand:
Sicherer Zustand = Das Laufzeitsystem deaktiviert alle sicheren Ausgänge. **Nicht** deaktiviert werden die für diesen Fehler mit ***SET_KEEP_ALIVE*** (→ Seite 386) ausgenommenen Ausgänge.
Bei schweren Fehlern ausschließlich an sicheren Ausgängen werden die CANsafety-Nachrichen weiterhin gesendet!

Programmablauf und Zykluszeit überwachen

13399

Der SafetyController hat eine interne Programmablauf-Überwachung. Diese Überwachung generiert (unter anderem) daraus das interne Watchdog-Signal (→ ***Watchdog*** (→ Seite 487)).

> Eine Testeinrichtung überwacht dieses Signal auf eine Periodendauer ≤ 100 ms.

> Bei Erkennen eines Ausfalles setzt die Testeinrichtung die Verarbeitung der Logik (CPU) in den sicheren Zustand (⇨ fataler Fehler).
  ℹ Sicherer Zustand = die Testeinrichtung deaktiviert die CPU und die Relais.
  ⇨ alle Ausgänge sind stromlos.

ℹ Die Laufzeitüberwachung erfolgt über die komplette Betriebssoftware, bestehend aus...

- Laufzeitsystem und
- Anwendungsprogramm.

Parameter von sicherheitsrelevanten FBs überwachen

13401

Beim Aufruf von sicherheitsrelevanten IEC-Funktionsbausteinen wird überprüft, ob die übergebenen Parameterwerte im zulässigen Bereich liegen.

Wird ein Fehler erkannt:

- eine Ausfallmeldung wird erzeugt
- die sicherheitsrelevanten Ausgänge werden in den sichern Zustand versetzt.

ℹ Sicherer Zustand = Das Laufzeitsystem deaktiviert alle als "safety" konfigurierten Ausgänge und stoppt das Senden von CANsafety-Nachrichten.

Sicherung nichtflüchtiger Daten

13400

Anwendungsdaten im Retain-Speicher

13840

Über die Anwendungsdaten im nichtflüchtigen Retain-Speicher wird vom System keine CRC gebildet, da das System nicht ermitteln kann, wann ein zulässiger schreibender Speicherzugriff erfolgt ist.

HINWEIS

- ► Sicherheitsrelevante Daten NICHT per **ifm**-Downloader aus dem Gerät auslesen oder in Geräte laden!
  Hierfür sind keine geeigneten Sicherungsmaßnahmen vorhanden.
- Sicherheitsrelevante Programme hingegen dürfen mit dem **ifm**-Downloader gelesen und kopiert werden.

► Der Programmierer muss daher mit geeigneten Methoden eine Absicherung der Daten per CRC-Signatur realisieren.
Dazu steht im Laufzeitsystem der FB ***CHECK_DATA*** (→ Seite 375) zur Verfügung.

Sicherung flüchtiger Daten im RAM

13788

Über die Anwendungsprogrammdaten (Variablen) im flüchtigen IEC-RAM wird vom System keine CRC gebildet, da das System nicht ermitteln kann, wann ein zulässiger Speicherzugriff erfolgt ist.

► Der Programmierer muss mit geeigneten Methoden eine Absicherung der Daten selbst realisieren.
Geeignete Methoden sind:
  - CRC-Signatur via FB ***CHECK_DATA*** (→ Seite 375) erzeugen und prüfen
  - Hamming-Code (z.B. für den Zustand einer Zustandsmaschine)
  - zyklisches Neuberechnen und Neuschreiben des Variablen-Wertes anhand aktueller Eingangswerte

► Der Programmierer muss im Fehlerfall einen "fatalen Fehler" mit dem FB ***ERROR_REPORT*** (→ Seite 382) an das System melden und so das System in einen sicheren Zustand bringen.
ℹ Sicherer Zustand = das Laufzeitsystem stoppt die Steuerung (fataler Fehler):
• alle Ausgänge werden abgeschaltet
• die Abarbeitung der Software wird angehalten
• es ist keine Kommunikation mehr möglich
• die LED erlischt.

► Für Neustart des Geräts:
• Power-On-Reset durchführen.

Systemdaten im Speicher

13841

Auf die Systemdaten hat der Anwender keinen direkten Zugriff.

Die sicherheitsrelevanten Systemdaten im nichtflüchtigen Speicher werden mit einer CRC-Signatur gesichert und beim Systemstart überprüft.

> Wird ein Fehler erkannt, geht das Gerät in den sicheren Zustand.

Sicherer Zustand = das Laufzeitsystem stoppt die Steuerung (fataler Fehler):
- alle Ausgänge werden abgeschaltet
- die Abarbeitung der Software wird angehalten
- es ist keine Kommunikation mehr möglich
- die LED erlischt.

Wird ein Fehler erkannt, während der TEST-Eingang aktiv ist:

> Teile der Systemdaten im Speicher werden auf sichere Default-Werte zurückgesetzt

> das Anwendungsprogramm und das Laufzeitsystem werden gelöscht

So kann wieder auf das Gerät zugegriffen werden.

Nicht-sicherheitsrelevante Systemdaten werden beim Erkennen eines Fehler auch ohne aktiven TEST-Eingang auf ihre Default-Werte zurückgesetzt.

Spannungen im System und der Ausgänge überwachen

13403

Der SafetyController überwacht zyklisch alle relevanten Spannungen im System.

> Wird ein schwerer Fehler (→ folgende Tabellen) erkannt:
  • eine Ausfallmeldung wird erzeugt
  • die sicherheitsrelevanten Ausgänge werden in den sichern Zustand versetzt.
  Sicherer Zustand = Das Laufzeitsystem deaktiviert alle sicheren Ausgänge und stoppt das Senden von CANsafety-Nachrichten.
  Nicht deaktiviert werden die für diesen Fehler mit ***SET_KEEP_ALIVE*** (→ Seite 386) ausgenommenen Ausgänge und CANsafety-Schnittstellen.

> Wird ein fataler Fehler (→ folgende Tabellen) erkannt:
  • die Steuerung wird in einen sicheren Zustand versetzt.
  Sicherer Zustand = das Laufzeitsystem stoppt die Steuerung (fataler Fehler):
  • alle Ausgänge werden abgeschaltet
  • die Abarbeitung der Software wird angehalten
  • es ist keine Kommunikation mehr möglich
  • die LED erlischt.

► Für Neustart des Geräts:
  • Fehlerursache beseitigen
  • Power-On-Reset durchführen.

Systemspannungen:

Potential	Spannungspegel	Dauer	Fehlermerker		Fehler / Fehlerklasse
VBBs	> 34 V	≥ 10 s	---	rot	fataler Fehler ¹)
VBBs	> 32 V	≥ 66 ms	ERROR_POWER = TRUE	rot	schwerer Fehler ²)
VBBs	> 10 V / < 31,5 V	---	---	grün	Normalbetrieb
VBBs	< 8 V	---	RETAIN_WARNING	grün	Normalbetrieb
VBBs	< ca. 7 V	---	---	gelb	Unterspannung VBBs
VBB15	VBBs + 1 V	≥ 66 ms	---	rot	fataler Fehler ¹)

Versorgungsspannungen der Ausgänge:

Potential	Spannungspegel	Dauer	Fehlermerker		Fehler / Fehlerklasse
VBBx	> 32 V	≥ 66 ms	ERROR_VBBx = TRUE	rot	schwerer Fehler ²)
VBBx	< 5,25 V	---	ERROR_VBBx = TRUE	rot	schwerer Fehler ²)
VBBx *)	< 5,25 V	---	ERROR_VBBx_E = TRUE	gelb	Unterspannung VBBx
VBBx	> 10 V / < 31,5 V	---	---	grün	Normalbetrieb
VBBrel	> 32 V	≥ 66 ms	ERROR_VBBx = TRUE	rot	schwerer Fehler ²)
VBBrel	< 4 V	---	ERROR_VBBx = TRUE	rot	schwerer Fehler ²)
VBBrel *)	< 4 V	---	ERROR_VBBrel_E = TRUE	gelb	Unterspannung VBBrel
VBBrel	> 10 V / < 31,5 V	≥ 66 ms	---	grün	Normalbetrieb

¹) Ein fataler Fehler kann nur durch Aus- und Einschalten des SafetyControllers zurückgesetzt werden.

²) Auch wenn der Spannungspegel wieder selbstständig in den zulässigen Spannungsbereich zurückkehrt, ist der Normalbetrieb nur möglich, wenn der Fehler-Code für den schweren Fehler zurückgesetzt wurde.

*) Extended-Seite im ExtendedSafetyController

3.3.6 Keep-Alive-Funktionalität

Inhalt

13419
1447

Auf einer Steuerung können mehrere unabhängige Sicherheitsfunktionen realisiert werden. Im Sinne der Verfügbarkeit ist es nicht immer gewollt und sinnvoll, dass beim Auftreten eines schweren Fehlers alle Sicherheitsfunktionen davon zugleich betroffen sind.

Keep-Alive stellt einen Mechanismus zur Verfügung, über den der Programmierer einzelne Ausgänge und CANsafety-Kanäle vom sicheren Zustand (= Ausgänge ausschalten) ausnehmen kann.

Keep-Alive kann nur angewendet werden...
- auf schwere Fehler und
- auf anwendungsspezifische Fehler.

Standardverhalten beim Auftreten eines schweren Fehlers

12332

Fehler, die im Laufzeitsystem erkannt werden, werden durch Ausfallmeldungen (Fehler-Codes) dem System mitgeteilt. Zyklisch prüft das Laufzeitsystem die Liste der Ausfallmeldungen.

Existiert in der Keep-Alive-Konfigurationsliste kein Eintrag für einen bestimmten Fehler-Code, so geschieht bei Auftreten dieses Fehlers Folgendes:
Das Laufzeitsystem deaktiviert alle sicheren Ausgänge und stoppt das Senden von CANsafety-Nachrichten.

Regeln für die Keep-Alive-Konfiguration

12333

HINWEIS

Bei 2-kanaliger Anschaltung beachten:
Bei einem Fehler in einem Kanal darf der andere Kanal nicht vom sicheren Zustand ausgenommen werden.

► Bei 2-kanaliger Anschaltung immer beide Kanäle gleichermaßen mit Keep-Alive behandeln!

Für ein Maximum an Verfügbarkeit kann die Anwendung durch eine Konfigurationsliste festlegen, wie auf jeden Fehler reagiert werden muss.

Jeder Fehler-Code, für den Keep-Alive gelten soll, muss konfiguriert werden.

Die Keep-Alive-Konfiguration erfolgt durch den Anwender im IEC-Anwendungsprogramm. Hierfür steht der FB ***SET_KEEP_ALIVE*** (→ Seite 386) zur Verfügung.

Ohne Keep-Alive-Konfiguration wird beim Erkennen eines schweren Fehlers das Standard-Verhalten auf die sicheren Ausgänge und CANsafety-Nachrichten angewendet.

Beim Aufruf des Funktionsbausteins zur Keep-Alive-Konfiguration werden die folgenden Parameter an das System übergeben:

- Fehler-Code (→ Kapitel ***Fehler-Codes*** (→ Seite 392))
- Ausgänge, die beim Auftreten des Fehler-Codes aktiv bleiben sollen
- CANsafety-Kanäle, die beim Auftreten des Fehler-Codes aktiv bleiben sollen

Empfohlene Reihenfolge im Anwendungsprogramm:

1.	Einem anwendungsspezifischen Fehler einen ERRORCODE zuordnen.
2.	Für jeden ERRORCODE (auch Systemfehler), bei dem nur ein Teil der Sicherheitsfunktionen betroffen ist, eine Keep-Alive Konfiguration erstellen.
3.	Beim Auftreten eines anwendungsspezifischen Fehlers den entsprechenden ERRORCODE dem System mitteilen. Bei Systemfehlern erfolgt dies automatisch.
4.	Wenn der Fehler beseitigt wurde, den entsprechenden ERRORCODE zurücksetzen

Regeln für den Fehler-Code

13687

Die hier folgenden Regeln gelten für den bei der Konfiguration übergebenen Fehler-Code.

Allgemein

13845

- Ein Fehler-Code wird nicht dahingehend überprüft, ob er im System auftreten kann.
- Es wird immer ein schwerer Fehler angenommen, unabhängig davon welche Fehlerklasse angegeben wird.

Konfiguration für einen Systemfehler

13846

Systemfehler sind Fehler, die vom Laufzeitsystem erkannt werden, z.B. Überspannung an der Versorgungsspannung.

- Es muss eine Fehlerursache und / oder eine Fehlerquelle angegeben werden.
  Der Anwendungsfehler muss den Wert 0 haben.
- Wird bei der Fehlerursache der Wert 0 angegeben, gilt die Konfiguration für alle Fehlerursachen bei der angegebenen Fehlerquelle.
- Wird bei der Fehlerquelle der Wert 0 angegeben, gilt die Konfiguration für die angegebene Fehlerursache, unabhängig davon, an welcher Fehlerquelle sie auftritt

Konfiguration für einen Anwendungsfehler

13847

Anwendungsfehler sind Fehler, die von der Anwendung definiert und erkannt werden und über den FB ***ERROR_REPORT*** (→ Seite 382) an das Laufzeitsystem gemeldet werden können.

- Fehlerursache und Fehlerquelle müssen den Wert 0 haben.
  Der Anwendungsfehler muss einen Wert > 0 haben.

Beispiele

13848

Konfiguration	Fehlerursache	Fehlerquelle	Anwendungsfehler	Fehlerklasse
Leiterbruch an Eingang I01	0x01	0x11	0x00	0x00
Überlast an allen Ein- und Ausgängen	0x04	0x00	0x00	0x00
alle Fehler an Eingang I10	0x00	0x1A	0x00	0x00
anwendungsspezifischer Fehler 20	0x00	0x00	0x14	0x00

Regeln für die Konfiguration

13688

Konfigurations-Timeout:
Eine Keep-Alive-Konfiguration darf nur innerhalb der ersten vier Sekunden nach Initialisierung der Steuerung vorgenommen werden. Ein späterer Konfigurationsversuch wird als schwerer Fehler gewertet.

Mehrfache Konfiguration:
Werden für denselben Fehler-Code mehrere Konfigurationen vorgenommen, kommt nur die letzte Konfiguration zur Anwendung.

Maximale Anzahl an Konfigurationen:
Die maximale Summe der gleichzeitig nutzbaren Konfigurationen (System- und Anwendungsfehler) beträgt 256.

Ändern des Standard-Verhaltens:
Wird bei der Konfiguration ein Fehler-Code mit den folgenden Werten angegeben, wird das Standard-Verhalten des Systems bei einem schweren Systemfehler geändert. Dann können sichere Ausgänge und CANsafety-Kanäle bei schweren Systemfehlern immer vom sicheren Zustand ausgenommen werden:

- Fehlerursache = 0
- Fehlerquelle = 0
- anwendungsspezifischer Fehler-Code = 0
- Fehlerklasse = 2 = schwerer Fehler

Das Ändern des Standardverhaltens hat keine Auswirkungen auf schwere Anwendungsfehler.

Konfigurationseinschränkungen:
Für folgende Fehler kann kein Keep-Alive konfiguriert werden:

- Parametrierfehler (Fehlerursache = 0xF8)

Keep-Alive-Verhalten bei einem Fehler

12341

Einmal im IEC-Zyklus wird überprüft, ob ein oder mehrere Fehler aufgetreten sind, die zu einem sicheren Zustand führen. Vor dem Herbeiführen des sicheren Zustands wird überprüft, ob es Keep-Alive-Konfigurationen zu diesen Fehlern gibt, welche bewirken, dass Ausgänge oder CANsafety-Kanäle von dem sicheren Zustand ausgenommen werden sollen:

- Liegt keine Konfiguration vor, wird das Standard-Verhalten für den sicheren Zustand angewendet.
- Liegt eine Konfiguration vor für einen Fehler-Code oder eine Fehler-Code-Gruppe (Fehlerursache=0 oder Fehlerquelle=0), wird diese Konfiguration angewendet.

Beim Auftreten eines oder mehrere Fehler prüft der Keep-Alive-Mechanismus, welche Konfigurationen auf den oder die Fehler anwendbar sind. Sind mehrere Konfigurationen gleichzeitig anwendbar, gilt folgende Regel:

- Ein sicherer Ausgang oder ein CANsafety-Kanal wird nur aktiv gehalten, wenn ALLE anwendbaren Konfigurationen dies so konfiguriert haben.
- Enthält eine der anwendbaren Konfigurationen keine Angabe zu einem Ausgang oder CANsafety-Kanal, wird für diesen Fehler der sichere Zustand eingenommen.
  → Kapitel ***Sicherer Zustand*** (→ Seite 52)
- Liegt für einen der aufgetretenen Fehler keine Konfiguration vor, wird das Standardverhalten für den sicheren Zustand angewendet.

ⓘ Der Fall, dass mehrere Konfigurationen auf einen Fehler angewendet werden können, kann bei der Verwendung von Fehler-Code-Gruppen vorkommen (Fehlerursache=0 oder Fehlerquelle=0).

Fehler signalisieren

12343

Es gibt zwei Varianten, wie Fehler an den Keep-Alive-Mechanismus gemeldet werden:

- Der Programmierer kann über den FB ***ERROR_REPORT*** (→ Seite 382) anwendungsspezifische Fehler-Codes an den Keep-Alive-Mechanismus übergeben.
- Das System (Laufzeitsystem) kann bei Erkennen eines Fehlers diesen Fehler direkt an den Keep-Alive-Mechanismus melden.

Fehler, die durch das Laufzeitsystem selbst erkannt werden, werden intern behandelt. Die dazugehörenden Fehler-Codes werden bei einigen FBs über einen Ausgang dem Anwendungsprogramm übergeben.

> Die aktuell aktiven Fehler-Codes auslesen:
  • über den Systemmerker ERRORCODE → Kapitel ***Fehler-Codes*** (→ Seite 392)
  • und den FB ***SHOW_ERROR_LIST*** (→ Seite 389)

Ausnahmen vom Keep-Alive

12344

In folgenden Fällen werden sichere Ausgänge oder CANsafety-Kanäle vom Keep-Alive ausgenommen, obwohl eine passende Konfiguration vorhanden ist:

- Der sichere Ausgang oder der CANsafety-Kanal ist selbst die Fehlerquelle eines anliegenden Fehlers.
- Der sichere Ausgang befindet sich in einer Ausgangsgruppe, die aufgrund eines Fehlers über den zweiten Abschaltweg deaktiviert wurde.
- Der sichere Ausgang befindet sich in einer Ausgangsgruppe mit zu niedriger Versorgungsspannung.

3.3.7 CANsafety im SafetyController

Inhalt

13418

Hier stellen wir Ihnen die CANsafety-Technologie vor, wie sie im **ecomatmobile**-SafetyController genutzt wird.

HINWEIS

- Sicherheitsrelevante Daten ausschließlich mit CANsafety übertragen!
  Andere CAN-Protokolle (z.B. CANopen, J1939) sind für sicherheitsrelevante Daten NICHT zugelassen.
- Mit anderen CAN-Protokollen übertragene Daten NICHT für sicherheitsrelevante Zwecke verwenden!

Allgemeine Hinweise und Erklärungen zu CANsafety

3848

Durch eine Arbeitsgruppe der Nutzerorganisation "CAN in Automation" (CiA) wurde eine Erweiterung des CANopen Kommunikationsprofils (CiA DS 304) erarbeitet. Diese ermöglicht parallel folgende Kommunikationsmöglichkeiten auf derselben Busleitung:

- "normale" Kommunikation zwischen CAN-Busteilnehmern (z.B. E/A-Module und einer Steuerung)
  Extended-CAN-Protokolle (29-Bit-ID, z.B. SAE J1939) dürfen NICHT auf dem CANsafety-Kanalpaar verwendet werden!
- sichere Daten zwischen Safety-CAN-Busteilnehmern austauschen.

Voraussetzung für diese gemeinsame Kommunikationsmöglichkeit: die Busteilnehmer, die diese sicheren Daten erzeugen oder lesen, müssen folgende Bedingungen einhalten:

- die Busteilnehmer unterstützen die entsprechenden Protokoll-Mechanismen und
- die Busteilnehmer weisen einen Aufbau der CAN-Hardware auf gemäß den Vorgaben des CANopen-Kommunikationsprofils.

Wie bei der Auslegung einer Sicherheitssteuerung und dem implementierten Anwendungsprogramm, muss auch bei einem "sicheren Bussystem" die Fehlerfreiheit der Daten gewährleistet werden. Tritt ein Fehler bei der Kommunikation auf, muss in einer hinreichend kurzen Zeit reagiert und die Maschine in einen sicheren Zustand gebracht werden.

Gleichzeitig dürfen die implementierten Sicherheitsfunktionen nicht die "normale" Kommunikation der nicht-sicheren Busteilnehmer negativ beeinflussen.

CANopen für die sichere Kommunikation

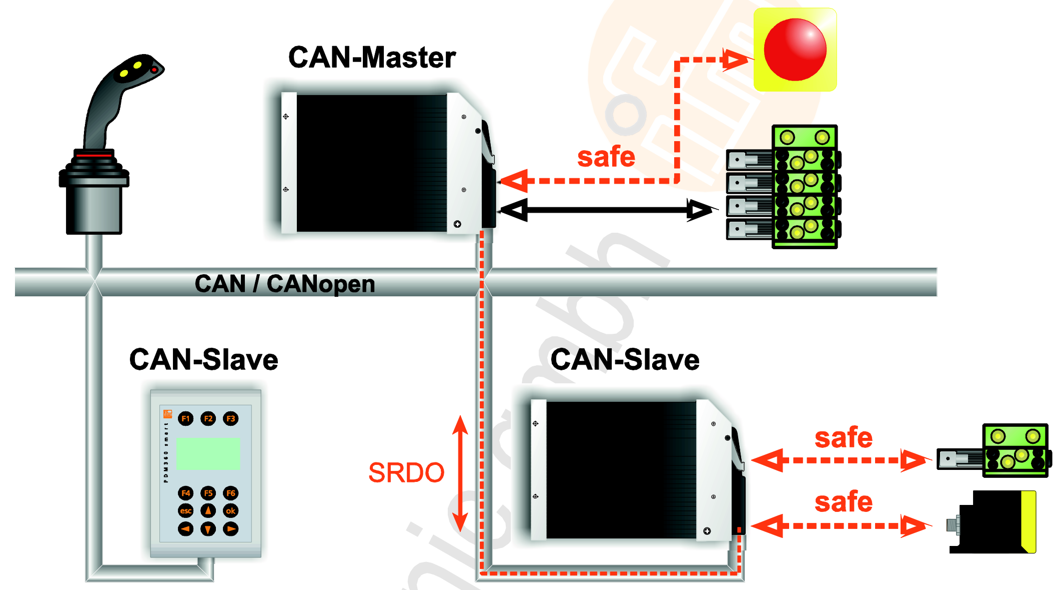
Inhalt

3851

Die gesamte "sichere" Kommunikation CANopen-Safety basiert auf den Standard-CAN-Mechanismen und ist in die CANopen-Kommunikationsprofile integriert. Dadurch ist es möglich, auf einer Busleitung gleichzeitig folgende Daten auszutauschen:

- "einfache" Daten zwischen nicht-sicheren Teilnehmern,
- "einfache" Daten zwischen nicht-sicheren und sicheren Teilnehmern,
- sichere Daten zwischen sicheren Teilnehmern.

! Extended-CAN-Protokolle (29-Bit-ID, z.B. SAE J1939) dürfen NICHT auf dem CANsafety-Kanalpaar verwendet werden!



Grafik: Parallele Kommunikation von normalen und sicheren CAN-Busteilnehmern

Die nachfolgend beschriebenen Dienste, Protokollmechanismen und Bausteine stellen eine Ergänzung zum Applikations- und Kommunikationsprofil CANopen dar. Diese CAN-Mechanismen garantieren nur den sicheren Austausch von Daten zwischen sicheren Busteilnehmern.

! Für die sichere Verarbeitung der Daten im Anwendungsprogramm ist allein dessen Programmierer verantwortlich.

Sicherheitsrelevante Datenobjekte SRDOs

3853

Über SRDOs (**S**afety-**R**elated **D**ata **O**bjects = ***Sicherheitsrelevante Datenobjekte*** (→ Seite 81)) werden bei CANsafety / CANopen Safety sichere Daten ausgetauscht. Ein SRDO besteht immer aus zwei →CAN-Nachrichten mit unterschiedlichen →Identifiern:
• Nachricht 1 enthält die Originalanwenderdaten,
• Nachricht 2 enthält die gleichen Daten, die aber bitweise invertiert werden.

Das Laufzeitsystem liest und vergleicht die Daten. Wurden die Daten fehlerfrei übertragen, stehen dem Anwendungsprogramm anschließend die Originaldaten zur Verfügung und können weiterverarbeitet werden.

Überwacht werden:

- bei der Übertragung verfälschte Daten
- zyklische Aktualisierung (SCT) der SRDOs (→ Kapitel ***Sicherheits-Zykluszeit SCT*** (→ Seite 81))
- der korrekte Abstand (SRVT) der redundanten Nachricht 2 zur Originalnachricht (→ Kapitel ***Sicherheitsrelevante Objekt-Gültigkeitsdauer SRVT*** (→ Seite 82))

Begrenzung der SRDOs

13395

Der SafetyController unterstützt für beide CANsafety-Schnittstellen zusammen bis zu 8 TX-SRDOs und 8 RX-SRDOs.

Die Anzahl der Teilnehmer, die diese Daten empfangen (=Consumer), ist nur durch die Netzwerkstruktur und die generellen CANopen-Mechanismen beschränkt.

Sicherheits-Zykluszeit SCT

3854

Bei CANsafety / CANopen Safety überprüft die Sicherheits-Zykluszeit SCT (**S**afeguard **c**ycle **t**ime) die korrekte Funktion der periodischen Übertragung (Daten-Refresh) der →SRDOs. Die Daten müssen innerhalb der eingestellten Zeit wiederholt worden sein, um gültig zu sein. Andernfalls signalisiert die empfangende Steuerung einen schweren Fehler und geht in den sicheren →Zustand (→ Kapitel ***Sicherer Zustand*** (→ Seite 52)).

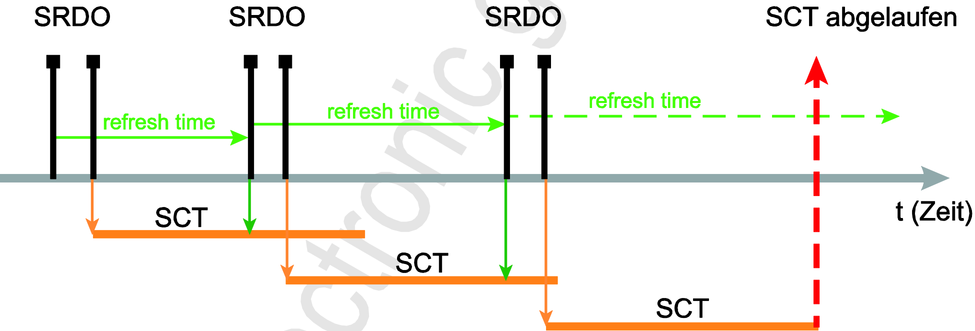


Bild: SCT überwacht den Daten-Refresh

Sicherheitsrelevante Objekt-Gültigkeitsdauer SRVT

3856

Die sicherheitsrelevante Objekt-Gültigkeitsdauer SRVT (**S**afety-**R**elated **O**bject **V**alidation **T**ime) sorgt bei CANsafety / CANopen Safety dafür, dass die Zeit zwischen den SRDO-Nachrichten-Paaren eingehalten wird:
Nur wenn die redundante, invertierte Nachricht innerhalb der eingestellten Zeit SRVT nach der Original-Nachricht übertragen wurde, sind die damit übertragenen Daten gültig. Andernfalls signalisiert die empfangende Steuerung einen schweren Fehler und geht in den sicheren →Zustand (→ Kapitel ***Sicherer Zustand*** (→ Seite 52)).

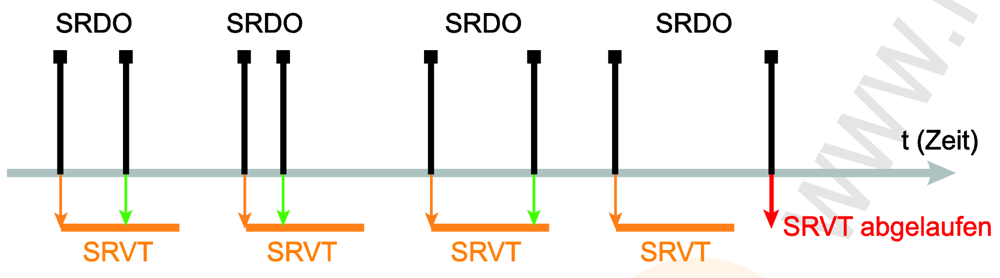


Bild: SRVT überwacht die Zeit zwischen den SRDO-Nachrichten

Global failsafe command GFC

3858

Um die Reaktionszeit des gesamten CANopen-Netzwerks zu erhöhen, kann das "**G**lobal **f**ailsafe **c**ommand" (GFC) gesendet werden. Die Nachricht ist ereignis-orientiert und nicht sicher. Das bedeutet, die Nachricht wird nur einmalig vom Producer versendet. Das GFC ist für alle Netzwerkteilnehmer gleich (gleicher Identifier) und enthält keine Daten. Man kann das GFC z.B. dafür nutzen, um eine Vorwarnung an alle sicherheitsgerichtete Teilnehmer im Netz zu schicken.

> Da das GFC nicht gesichert übertragen wird, darf es nicht mit in die Berechnung der Erstfehler-Eintrittszeit einbezogen werden.

Verarbeitung der SRDO im SafetyController

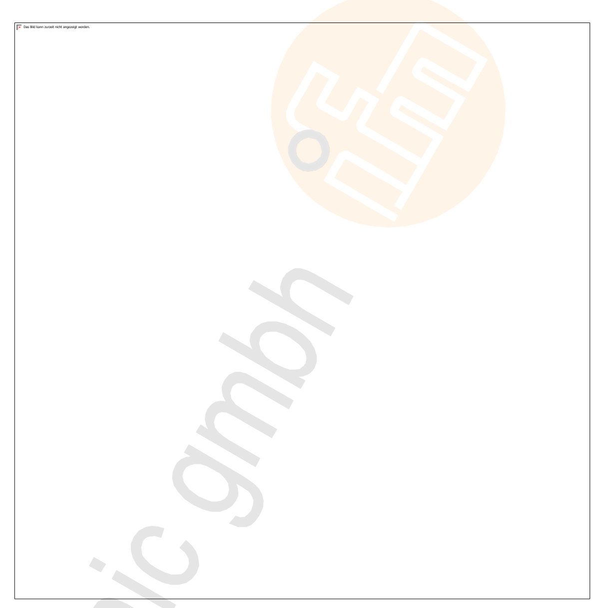
13689

Um die SRDO-Daten im SafetyController sicher zu verarbeiten, müssen diese Daten **über 2 CAN-Schnittstellen** eingelesen werden. Durch die Unterschiede in der Hardware- und Software-Schnittstelle wird gewährleistet, dass die übertragenen Daten fehlerfrei dem nachfolgenden Anwendungsprozess zugeführt und weiterverarbeitet werden können.

In einem CAN-Bussystem, in dem auch sichere Daten übertragen werden sollen, wird die Busleitung mit beiden CAN-Schnittstellen verbunden. Durch die Anwendung von ***CAN_SAFETY_TRANSMIT*** (→ Seite 217) und ***CAN_SAFETY_RECEIVE*** (→ Seite 214) werden die Daten über beide Schnittstellen eingelesen und gemäß des "CANopen framework for safety-relevant communication" DS 304 geprüft und an das Anwendungsprogramm übergeben.

Weitere zusätzliche Protokolle (z.B. CAN Layer 2 oder CANopen) werden "parallel" zu CANsafety übertragen und verarbeitet.
Empfehlung: für alle nicht-sicheren Protokolle vom CANsafety-Kanalpaar die erste CAN-Schnittstelle verwenden (⇨ CAN1, CAN3)!

Extended-CAN-Protokolle (29-Bit-ID, z.B. SAE J1939) dürfen NICHT auf dem CANsafety-Kanalpaar verwendet werden!



Grafik: CANsafety-Teilnehmer benötigen 2 CAN-Schnittstellen für sichere Datenübertragung

Da es sich beim SafetyController um ein frei programmierbares Gerät handelt und bedingt durch den internen Aufbau der implementierten unterschiedlichen CANopen-Schnittstellen, wurde für den Austausch der CANsafety-Parameter kein Objektverzeichnis (OBV) implementiert.

Alle Einstellungen, Netzwerkkommandos und die Datenübertragung werden mittels ***CAN_SAFETY_TRANSMIT*** (→ Seite 217) und ***CAN_SAFETY_RECEIVE*** (→ Seite 214) abgewickelt.

Das bedeutet, ein externes Konfigurations-Tool oder ein CANopen-Master kann die Parameter NICHT per SDO einstellen.

Vordefinierte Identifier für CANsafety

3861

Im Normalfall wird in CANopen-Netzwerken mit vordefinierten Identifiern gearbeitet. Für CANsafety wurde ebenfalls ein Identifier-Bereich festgelegt. Auch dieser ist in das "pre-defined connection set" von CANopen integriert.

HINWEIS

Es wird dringend empfohlen, nur die Identifier aus dem "pre-defined connection set" zu verwenden, da andernfalls der Überblick über das Netzwerk verloren geht.

► Der Programmierer muss sich in jedem Fall von der Konfliktfreiheit der CAN-Nachrichten überzeugen.

Wie oben beschrieben, besteht ein SRDO immer aus einem Nachrichten**paar**. Nachricht 1 enthält die regulären Daten und einen Identifier mit einem ungeraden Wert. Nachricht 2 enthält die invertierten Daten und einen Identifier mit geradem Wert.

Das "pre-defined connection set" geht davon aus, dass jeder sichere Teilnehmer eine Transmit- und eine Receive-Nachricht versendet.

Der SafetyController unterstützt für beide CANsafety-Schnittstellen zusammen bis zu 8 TX-SRDOs und 8 RX-SRDOs. Die SRDOs sind nur im folgenden Identifier-Bereich zulässig:

Objekt	CAN-Identifier			
	reguläre Daten		invertierte Daten	
	dez.	hex.	dez.	hex.
TX-SRDO SafetyController 1 RX-SRDO SafetyController 2	257	101	258	102
	259	103	260	104
	261	105	262	106
	...	...	...	...
	319	13F	320	140
RX-SRDO SafetyController 1 TX-SRDO SafetyController 2	321	141	322	142
	323	143	324	144
	325	145	326	146
	...	...	...	...
	383	17F	384	180

Beispiel:

Für SafetyController 1 soll gelten:
- für das Sende-SRDO sei die Identifier-Kombination 257 und 258
- für das Empfangs-SRDO sei die Identifier-Kombination 321 und 322.

Dann gilt für den korrespondierenden SafetyController 2:
- das Empfangs-SRDO hat die Identifier-Kombination 257 und 258
- das Sende-SRDO hat die Identifier-Kombination 321 und 322.

3.3.8 Zertifizierte Software-Bausteine für sicherheitsrelevante Anwendungen

13428

ifm bietet passend zu diesem Gerät eine Reihe von zertifizierten Software-Bausteinen. Diese Bausteine erleichtern das Programmieren von sicherheitsrelevanten Anwendungen.
In der folgenden Tabelle finden Sie die Bausteine zusammen mit einer kurzen Erläuterung:

Baustein	Kurzbeschreibung
CAN_SAFETY_RECEIVE (→ Seite 214)	empfängt eine sichere CAN-Nachricht (SRDO)
CAN_SAFETY_TRANSMIT (→ Seite 217)	überträgt eine sichere CAN-Nachricht (SRDO)
SAFETY_SWITCH (→ Seite 268)	Betrieb der 1-kanaligen SafetySwitch der **ifm electronic gmbh**
SF_ANTIVALENT (→ Seite 272)	• vergleicht zwei binäre sichere Eingänge miteinander • prüft den zeitlichen Ablauf der Eingänge zueinander
SF_EMERGENCYSTOP (→ Seite 274)	überwacht einen Not-Halt-Schalter
SF_ENABLESWITCH (→ Seite 277)	dient zum Auswerten der Signale einer Freigabetaste mit drei Schaltstufen
SF_ENABLESWITCH_2 (→ Seite 280)	dient zum Auswerten der Signale einer Freigabetaste mit zwei oder drei Schaltstufen
SF_EQUIVALENT (→ Seite 283)	• vergleicht zwei binäre sichere Eingänge miteinander • prüft den zeitlichen Ablauf der Eingänge zueinander
SF_EQUIVALENT_REAL (→ Seite 285)	• vergleicht zwei sichere Eingangswerte [REAL] miteinander • prüft die Werte auf zulässigen Wertebereich und zulässige Abweichung
SF_EQUIVALENT_WORD (→ Seite 287)	• vergleicht zwei sichere Eingangswerte [WORD] miteinander • prüft die Werte auf zulässigen Wertebereich und zulässige Abweichung
SF_MODESELECTOR (→ Seite 199)	ermöglicht das sichere Schalten zwischen bis zu 8 Betriebsarten einer Maschine oder Anlage
SF_OUTCONTROL (→ Seite 319)	kontrolliert einen sicheren Ausgang mit einem Signal aus der funktionellen Anwendung und einem sicheren Signal mit optionaler Anlaufsperre
SF_SAFETYREQUEST (→ Seite 322)	stellt eine Schnittstelle zu einem allgemeinen Aktuator zur Verfügung, um den Aktuator in den sicheren Zustand zu setzen
SF_TWOHANDCONTROL (→ Seite 289)	realisiert eine Zweihandbedienung

3.4 Beispiel: sichere Steuerung für eine Hubarbeitsbühne

Inhalt

13286

Auf Basis des nachfolgenden Beispiels (für einen Teil der Funktionen) soll der Ablauf bei der Risikoabschätzung vorgestellt werden. Diesen Ablauf muss der Maschinenhersteller für jede einzelne Sicherheitsfunktion durchlaufen.



Grafik: Sicherheitsparameter an einer Hubarbeitsbühne

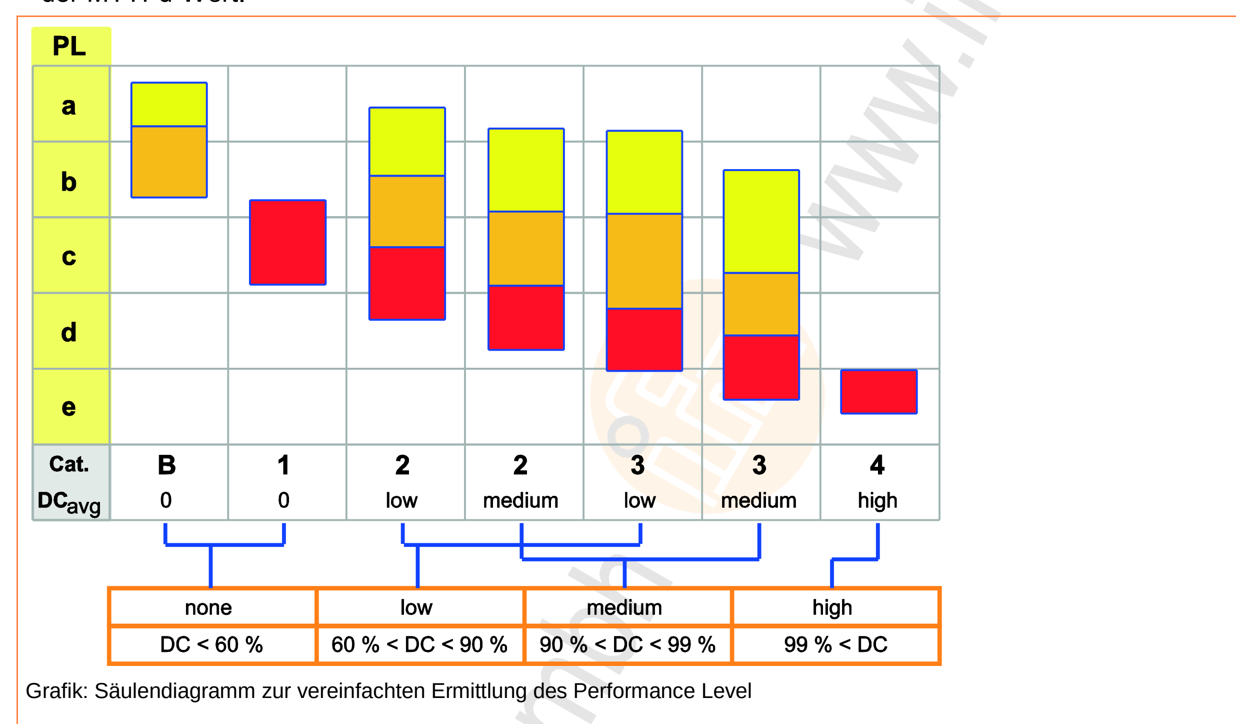
Legende:
(1) zulässige Last auf der Bühne
(2) zulässige Ausfahrlänge des Auslegers
(3) zulässiger Steigungswinkel des Auslegers
(4) zulässiger Drehwinkel des Auslegers
(5) abgesenkte Stabilisatoren
(6) Fahrzeug steht waagerecht
(7) geschlossenes Schutzgitter auf der Bühne

3.4.1 Säulendiagramm

13287

Um den möglichen Performance Level der einzelnen Sicherheitsfunktionen möglichst einfach zu ermitteln, bietet sich das vereinfachte Verfahren mit dem Säulendiagramm an. Aus dem Diagramm kann sehr einfach der mögliche PL abgelesen werden, wenn folgende Parameter vorliegen:

- die Kategorie (Cat.),
- der Diagnosedeckungsgrad (DC) und
- der MTTFd-Wert.



Grafik: Säulendiagramm zur vereinfachten Ermittlung des Performance Level

Legende MTTFd:

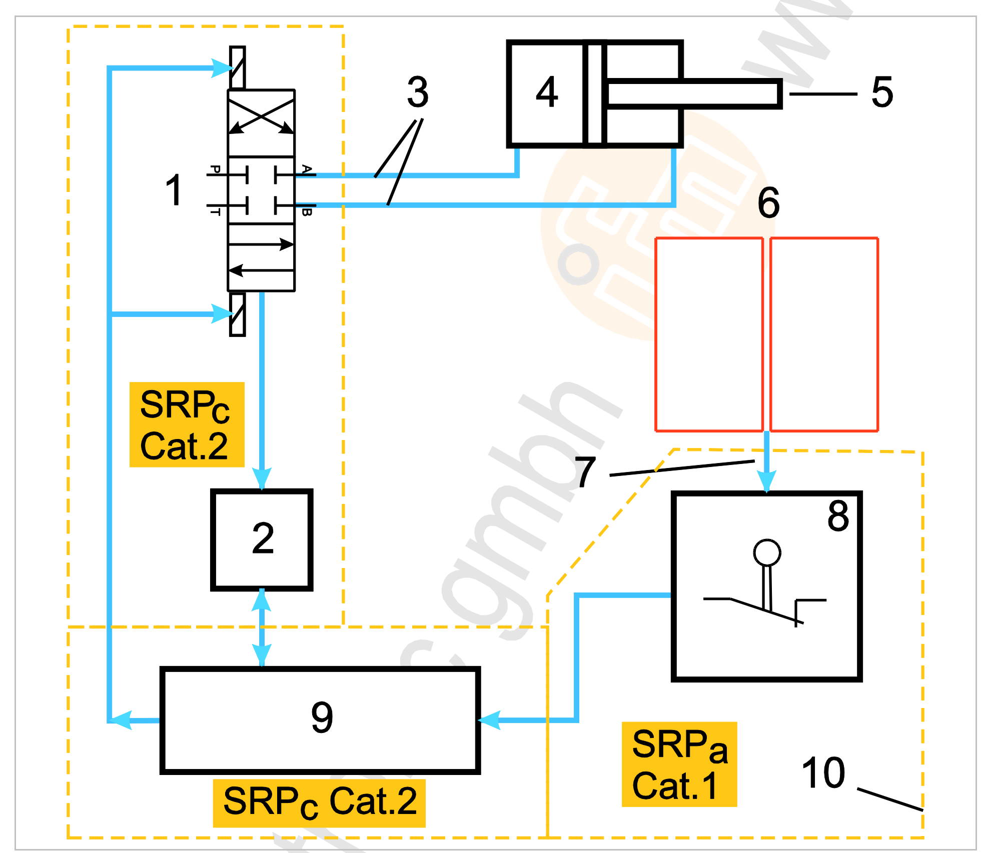
	low	3 Jahre < MTTFd < 10 Jahre
	medium	10 Jahre < MTTFd < 30 Jahre
	high	30 Jahre < MTTFd < 100 Jahre

3.4.2 Beispiel aus EN 280 (Kap. 5.11): Sicherheitseinrichtung

13448

Beispielbeschreibung:

- Die elektronische Steuerlogik (9) gibt – abhängig von Situation und Programm – Stellsignale an das hydraulische Wegeventil (1).
- Eine Prüffunktion (2) vergleicht die Istposition des Ventils (1) mit der Sollposition und meldet das Ergebnis an die elektronische Steuerlogik (9) zurück.
- Ein Hydraulikzylinder (4) kann sich gefährdend (5) bewegen.
- Eine trennende Schutzeinrichtung (6) schützt den Bediener vor der gefährdenden Bewegung (5).
- Eine Stellungsüberwachung (8) meldet die Position der trennenden Schutzeinrichtung (6) an die elektronische Steuerlogik (9) zurück.



Legende:

1	hydraulisches Wegeventil
2	Stellungsüberwachung des Ventils
3	Ausgangssignal zum hydraulischen Zylinder
4	hydaulischer Zylinder
5	gefährdende Bewegung
6	trennende Schutzeinrichtung
7	Eingangssignal von der trennenden Schutzeinrichtung
8	Stellungsüberwachung der trennenden Schutzeinrichtung
9	elektronische Steuerlogik
10	Anwendungsbereich der ISO 13849-1
CAT	Kategorie der Maschinenfunktion (Sicherheitsfunktion)
SRP	**S**afety-**R**elated **P**art = sicherheitsrelevantes Teil

3.4.3 Arbeitsschritte

13290

Schritt	Tätigkeit	Hinweis
1.	► Erforderlichen Performance-Level PLr für die Maschinenfunktion bestimmen. → ISO 13849, Anhang A oder: → Produktnorm ⇨ Für diese Anwendung ist der erforderliche Performance-Level c und d.	aus Norm EN 280
2.	► Sicherheitskonzept entwerfen ► sicherheitsrelevante Teile identifizieren, die die Sicherheitsfunktion ausführen ► die Sicherheitsfunktion technisch realisieren Beispiel: Umkippen der Maschine beim Hebevorgang: ⇨ Einschränkung der Reichweite im Betrieb ohne abgesenkte Stabilisatoren (EN 280 ⇨ PLr = PL c) Beispiel: Stabilitätsverlust / unkontrollierte Bewegungen während des Hebevorgangs: ⇨ Lastmomentbegrenzung (EN 280 ⇨ PLr = PL d)	aus Norm EN 280
3.	► Wahl der Sensoren, des SafetyController und der Aktoren ► Systemdesign erstellen unter Berücksichtigung der ausgewählten Hardware ► Anwendungsprogramm erstellen Beispiele: • Sicherheitssensoren wählen (z.B. ifm GG712S) • E/A-Konfiguration festlegen • Software-Design für Sicherheitsfunktion erstellen	
4.	► Wenn die Angaben nicht vorliegen, den Performance-Level PL der obigen sicherheitsrelevanten Teile und Funktionen mit Hilfe des Säulendiagramms bestimmen. Dabei berücksichtigen: • hier: die geforderte Kategorie (Cat.) der Produktnorm EN 280 entnehmen (dort gefordert: 1-kanalig mit Selbsttest (⇨ Cat. 2)) und daraus folgend: • die mittlere Lebensdauer MTTFd → Datenblatt Hersteller • den Diagnose-Deckungsgrad DC → Datenblatt Hersteller ► Den möglichen Ausfall in Folge von gemeinsamer Ursache CCF speziell bei Cat. 2...Cat. 4 beachten!	aus Norm EN 280
5.	► PL für die gesamte Sicherheitsfunktion bestimmen durch einen vereinfachten Ansatz der Reihenschaltung (→ folgende Tabelle). Dabei bestimmt der niedrigste PL-Wert ($PL_{niedrig}$) und die Anzahl der Bauteile mit diesem Wert den maximal erzielbaren PL. Hier: bei der Ausfallwahrscheinlichkeit den mittleren Wert des jeweiligen $PL_{niedrig}$ annehmen.	



Grafik: Vereinfachte Darstellung einer Sicherheitsfunktion
Hier: das Ventil hat mit PL c den $PL_{niedrig}$

$PL_{niedrig}$	Anzahl Bauteile mit $PL_{niedrig}$	resultierender PL
a	≥ 4	kein PL (nicht erlaubt)
	≤ 3	a
b	≥ 3	
	≤ 2	b
c	≥ 3	
	≤ 2	c
d	≥ 4	
	≤ 3	d
e	≥ 4	
	≤ 3	e

Tabelle: vereinfachte PL-Bestimmung für in Reihe geschaltete Bauteile

Schritt	Tätigkeit	Hinweis
6.	Die Überprüfung muss zeigen, dass die Gerätekombination für jede Sicherheitsfunktion die entsprechenden Anforderungen der Produktnorm oder der Risikoanalyse erfüllt. Wenn alle Anforderungen erfüllt: ▶ weiter mit Schritt 7 Wenn nicht alle Anforderungen erfüllt: ▶ zurück zum Schritt 2	
7.	Wenn alle Sicherheitsfunktionen analysiert: ⇨ Fertig! Sonst: ▶ nächste Sicherheitsfunktion analysieren	

3.5 Regeln für sicherheitsrelevante Anwendungen

Inhalt

13294

3.5.1 Regel 1 – Einbau und Verdrahtung der Sicherheitssteuerung

13295

► Insbesondere zu folgenden Punkten die →Montageanleitung des Geräts beachten!
• Einbaulage
• Montage
• Anschluss
• Absicherungen

⚠ WARNUNG

Bei Rückspeisung besteht Gefahr, dass der Ausgang nicht mehr deaktiviert werden kann!

Bei der Verwendung eines gegen Masse schaltenden Binärausgangs, kann es zu einer Rückspeisung in die Versorgung der Ausgänge kommen. Hierbei fließt der Strom über die angeschlossene Last und die im highside-Switch befindliche parasitäre Schutzdiode in den Versorgungsstrang zwischen dem Relais und den Ausgängen dieser Gruppe zurück.

Selbst bei der Abschaltung über das Relais (zweiter Abschaltweg) kann dadurch bei einem defekten Ausgang eine Spannung anliegen.

► Die Last an einem gegen Masse schaltenden Ausgang nur über einen gegen Versorgung schaltenden Ausgang derselben Ausgangsgruppe versorgen!

Verzögertes Abschalten der Ausgänge über das Relais (zweiter Abschaltweg) möglich, falls sichere Ausgänge kapazitive Lasten schalten sollen. Dadurch könnte die Sicherheitszeit nicht eingehalten werden.

► Bei der Verwendung von kapazitiven Lasten am sicheren Ausgang:
eine Diode in Flussrichtung schalten!

→ Kapitel ***Rückspeisung bei extern beschalteten Ausgängen*** (→ Seite 148)

Querschluss vermeiden

13392

► Im 2-kanaligen Betrieb Querschlüsse zwischen zwei Eingangskanälen ausschließen! Im Sicherheitskonzept die Trennung der Eingangssignale voneinander vorsehen:
• beide Eingangssignale auf getrennte Eingangsblöcke konfigurieren!
Beispiel:
• Signal 1 an einem der Eingänge I00…07,
• Signal 2 an einem der Eingänge I08…15.

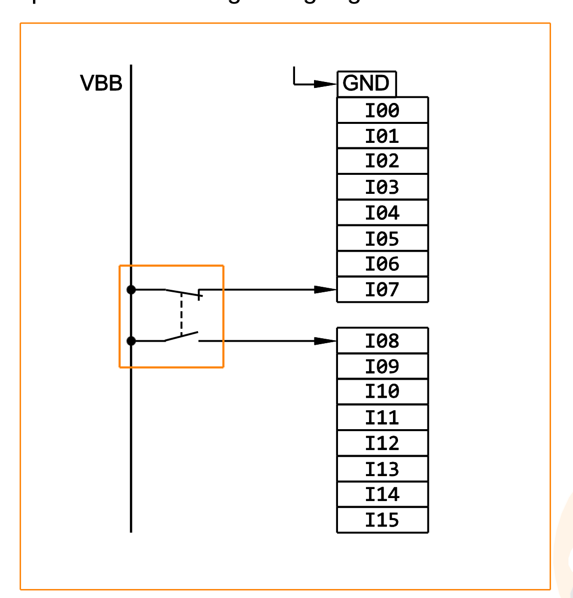
► Im 2-kanaligen Betrieb Querschlüsse zwischen zwei Ausgangskanälen ausschließen! Im Sicherheitskonzept die Trennung der Ausgangssignale voneinander vorsehen:
• beide Ausgangssignale auf getrennte Ausgangsblöcke konfigurieren! Dabei sind nur asymmetrische Beschaltungen erlaubt.
Beispiele:
• zulässig: Signal 1 an Ausgang Q00, Signal 2 an Ausgang Q09
• verboten: Signal 1 an Ausgang Q00, Signal 2 an Ausgang Q08.

► Querschlüsse in Kabeln zu den Ein- und Ausgängen vermeiden! Dazu bewährte Sicherheitsprinzipien einsetzen, z.B.:
- Je Kanal einzelne Leitungen verwenden, deren Abschirmung mit dem Schutzleitersystem verbunden ist, oder:
- Anwenden eines ausreichenden Abstands zwischen den Leitungen, so dass unbeabsichtigte Verbindungen vermieden werden, oder:
- Leitungen vor mechanischen Beschädigungen schützen (z.B. über ein Schutzrohr).

Anschlussbeispiele: sicherheitsrelevante 2-kanalige Signale

13132

Beispiele für 2-kanalige Eingänge:

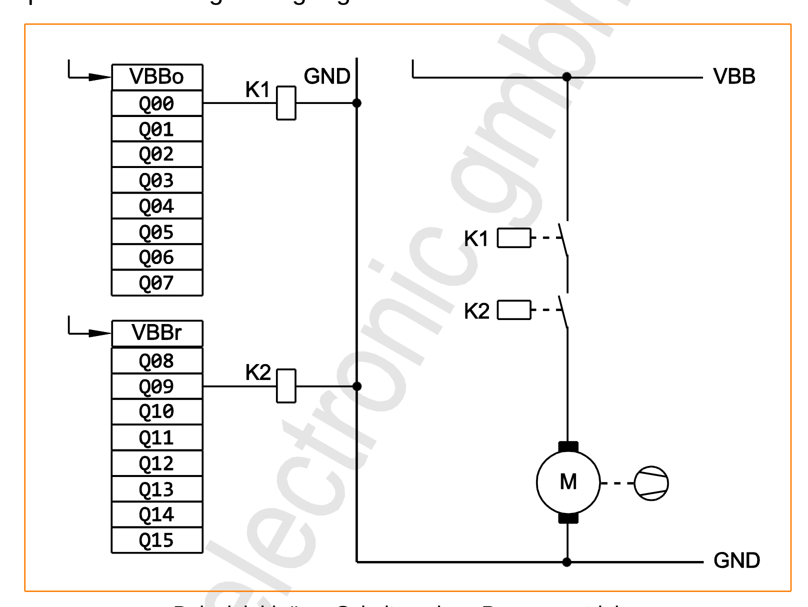


Beispiel: binärer Sicherheitssensor:
Eingang I07 mit Öffnersignal,
Eingang I08 mit Schließersignal
desselben Sensors.

VBB
GND
I00
I01
I02
I03
I04
I05
I06
I07
I
I08
I09
I10
I11
I12
I13
I14
I15
U

Beispiel: analoger Sicherheitssensor:
Eingang I04 mit Stromsignal,
Eingang I10 mit Spannungssignal
desselben Sensors.

Beispiel für 2-kanalige Ausgänge:



Beispiel: binäres Schalten eines Pumpenantriebs:
Ausgang Q00 für Relais K1,
Ausgang Q09 für Relais K2,
Kontakte von K1 und K2 schalten in Reihe den Pumpenantrieb.
▶ Schaltzustände von K1 und K2 überwachen!

Schutz gegen Überspannung

13460

Grundsätzlich sind alle Spannungseingänge durch den Aufbau der Hardware gegen Überspannung geschützt.

► Den SafetyController nur über das Bordnetz der mobilen Arbeitsmaschine (12 / 24 V DC) versorgen!

► Die Sensoren über das gleiche Bordnetz versorgen wie den SafetyController!

Erfolgt stattdessen eine Versorgung über ein Netzteil, darf die Netzteil-Ausgangsspannung auch im Fehlerfall den zulässigen Versorgungsspannungsbereich des SafetyControllers nicht überschreiten!
Diese Netzteile sind nur im Labor zulässig, nicht in der Anwendung in der Maschine!

3.5.2 Regel 2 – Schutz vor unbefugtem Zugriff

14462

Der **ecomat*mobile***-SafetyController besitzt keine integrierten Sicherheitsmechanismen (Zugriffsrechte, verschlüsselte Datenübertragung, ...), mit denen ein unerlaubter Zugriff über die Service-Schnittstellen verhindert werden könnte. Daher:

HINWEIS

► Die in der Maschine/Anlage montierten **ecomat*mobile***-SafetyController vor unerlaubtem Zugriff über die Service-Schnittstellen schützen.
Das gilt in besonderem Maße bei der Realisierung einer Fernwartungsmöglichkeit.

Geeignete Maßnahmen können z.B. sein:
- Passwortweitergabe ausschließlich an autorisierte Personen
- keine Anbindung der Steuerung an Netzwerke mit öffentlichem Zugang (z.B. Internet)

3.5.3 Regel 3 – Spezifikation des Sicherheitsprogramms

13298

Hauptziel bei der Erstellung von einem sicherheitsrelevanten Anwendungsprogramm ist:
⇨ lesbare, verständliche, wartbare und fehlerfreie Software.

Dafür muss gemäß ISO 13849 eine Spezifikation erstellt werden, die die folgenden Elemente enthält:

- Alle Sicherheitsfunktionen beschreiben mit erforderlichem PLr und zugehörigen Betriebsarten.
- Kriterien, die die Sicherheitsfunktionen erfüllen müssen, z.B. Reaktionszeiten.
- System-Architektur darstellen und die einzelnen Signalschnittstellen beschreiben.
- Methoden zum Erkennen und Beherrschen der Fehler und gefährlichen Fehler entwickeln.
- Mit einem nachvollziehbaren Verfahren die Verarbeitung der Programmdaten und den Programmablauf beschreiben (z.B. Zustandsdiagramm oder Programmflussdiagramm).

→ Kapitel ***Mit dem V-Modell das Erstellen der sicheren Maschine organisieren*** (→ Seite 29)

HINWEIS

► Die Spezifikation für jede Person verfügbar machen, die am Lebenszyklus der Software beteiligt ist.

► Das Anwendungsprogramm grundsätzlich unter Beachtung aller Informationen und der Hinweise aus der Gerätedokumentation erstellen!

3.5.4 Regel 4 – Sicherheitsrelevante Software dokumentieren

13299

- ► Alle Lebenszyklus- und Änderungsaktivitäten dokumentieren.
- ► Die Dokumentation muss verfügbar, vollständig, lesbar und verständlich sein.
- ► Die Dokumentation des Quellcodes sollte am Beginn jedes Funktions- und Programmbausteine folgende Angaben enthalten:
  - Name des Programmierers,
  - Beschreibung der Funktion und der Ein- und Ausgänge,
  - Version der verwendeten FB-Bibliothek,
  - ausreichend dokumentierte Netzwerke,
    ausreichend dokumentierte Anweisungen,
    ausreichend dokumentierte Deklarationszeilen,
  - Versionshistorie.
- ► Zusätzlich sollte einmal im Anwendungsprogramm enthalten sein:
  - die verwendete Laufzeitsystem-Version,
  - die Programmier-Software
  - die eingesetzte Hardware.

3.5.5 Regel 5 – Wahl der Sprachen und Bibliotheken

13300

Prinzipiell können für die Erstellung des Anwendungsprogramms alle Programmiersprachen von CODESYS eingesetzt werden. Jedoch:

- ► Zum Verbessern von Lesbarkeit und leichterer Überprüfung des Software-Codes folgende Programmiersprachen bevorzugen:
  - Funktionsplan "FUP" (function block diagram "FBD")
  - Kontaktplan "KOP" (ladder diagram "LD")
  - Strukturierter Text "ST" (structured text "ST")

Software-Bibliotheken ...
- erleichtern das Erstellen des Anwendungsprogramms
- minimieren Programmierfehler

Diese Software-Bibliotheken sollten unbedingt eingesetzt werden. Dies sind insbesondere:
- CODESYS-Funktionsbibliotheken (z.B. `Standard.lib`)
- ifm-Geräte-Bibliotheken (z.B. `ifm_CR7032.lib`)
- von ifm electronic verfügbare Funktionsbibliothek nach PLCopen (z.B. `ifm_SafetyPLCopen.lib`)

(!) Bei Verwendung anderer Bibliotheken die Eignung für den Einsatz in Sicherheitsfunktionen prüfen!

3.5.6 Regel 6 – Regeln zum Aufbau des Anwendungsprogramms

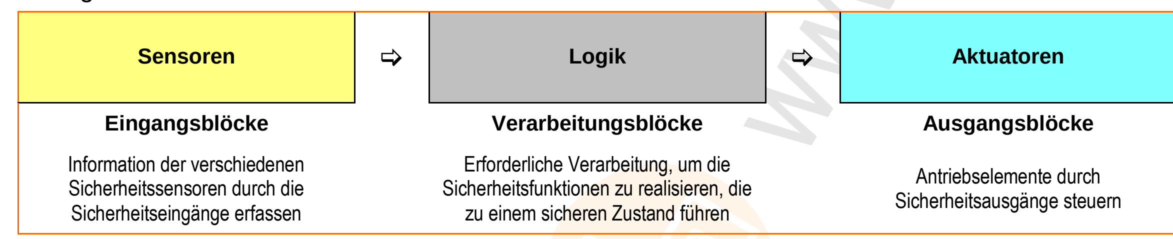
Inhalt

13301

Allgemeines

13302

- Das Anwendungsprogramm mit folgenden Zielen programmieren:
  - die Maschine und die sicherheitsrelevanten Teile und Funktionen sind einfach zu verwenden,
  - der Benutzer wird nicht zu einer Manipulation verleitet
    (z.B. Sicherheitseinrichtungen überbrücken oder demontieren).
- Das Anwendungsprogramm modular aufbauen und klar strukturieren.
- Als erste Orientierung sollte der Programmaufbau dem folgenden Modell folgen:
  Architektur des Modells in 3 Stufen: Eingänge ⇨ Verarbeitung ⇨ Ausgänge.
  → z.B. ISO 13849, Anhang J
  → folgende Grafik:



> Dieser Aufbau...
• ermöglicht eine einfache Testbarkeit
• ermöglicht ein leichteres Auffinden der einzelnen Programmfunktionen
• verbessert die Lesbarkeit bei der Bildschirmnavigation
• verbessert die Lesbarkeit des späteren Dokumentationsausdrucks.
• ermöglicht dadurch die spätere Programmerweiterung und -anpassung.

- Die Funktions- und Programmbausteine im Quellcode des Anwendungsprogramms dokumentieren:
  • die Beschreibung
  • das Verhalten
  • die Rückgabewerte
  • die Fehlercodes.
- Als Programm-Basis die von **ifm electronic** zur Verfügung gestellten Templates verwenden.
- Einfache, nicht verschachtelte Funktionen verwenden.
  ⇨ keine indirekte Adressierung
- Möglichst oft in der Entstehungsphase das Anwendungsprogramm testen.
  ⇨ Fehler werden so frühzeitig erkannt und eliminiert.
- Im Kommentar übersichtlich und ausführlich beschreiben:
  • die Aufgabe und Bedeutung der Variablen
  • die Aufgabe und Bedeutung der Funktionen
  • die Aufgabe und Bedeutung des sonstigen Anwendungsprogramms
  • Korrekturen und Programmergänzungen
  ⇨ Dieses erleichtert die Prüfung erheblich.
- Mit FB ***SET_IDENTITY*** (→ Seite 379) dem Anwendungsprogramm einen Namen und eine Versionsangabe mitgeben.
  Diese Daten nach der Übertragung auf das Gerät mit dem Downloader lesen und prüfen.
- Im Anwendungsprogramm mit FB ***GET_IDENTITY*** (→ Seite 377) durch Auslesen der Laufzeitsystem-Version prüfen, dass das richtige Laufzeitsystem geladen wurde.
  Im Fehlerfall das Anwendungsprogramm mit Fehlermeldung anhalten!

Programmstruktur

13303

- Die komplette Anwendung und die Programmmodule sollten vom Programmbaustein PLC_PRG aus aufgerufen werden. Im PLC_PRG sollte sonst nichts programmiert werden (also keine Logikverarbeitung).
- Umfangreiche Initialisierungen gegebenenfalls auf mehrere Programmzyklen verteilen.
- Sicherheitsfunktionen in eigenen, klar gekennzeichneten CODESYS-Programm- und Funktionsbausteinen von den übrigen Steuerungsfunktionen getrennt realisieren.
- Wenn möglich, für Sicherheitsfunktionen die von ifm zur Verfügung gestellten Bibliotheken nutzen. Diese wurden auf Basis der geltenden Normen entwickelt, geprüft und durch eine unabhängige Stelle zertifiziert.
- Eigene sichere Funktionsbausteine (FBs) in einem möglichst einfachen und übersichtlichen Code schreiben.
- Sicherheitsfunktionen sollen keine nicht-sicheren Funktionen aufrufen.
  Dieses in CODESYS mit der Programmfunktion [Projekt] > [Aufrufbaum ausgeben] überprüfen.
- Nicht-sichere und sichere Daten nicht logisch zu einem sicheren Ausgangssignal verknüpfen!
  Dies führt zur Herabstufung der Integrität des sicheren Ausgangssignals.

> **HINWEIS**
>
> Eine CSV-Datei darf keine sicherheitsrelevanten Daten enthalten.
> Hierfür sind keine geeigneten Sicherungsmaßnahmen vorhanden.

- Jeden Sicherheitsausgang nur jeweils **einem** Programmteil zuordnen.
  Keine Zuweisungen in mehreren Programmteilen!
- Die vom SafetyController zur Verfügung gestellten Diagnosefunktionen im Anwendungsprogramm verarbeiten und wenn notwendig darauf reagieren.

CODESYS-Projekt mit Passwort sichern

14169

- Den Zugriff von nicht-autorisierten Personen auf die Software verhindern!
  Dazu das CODESYS-Projekt mit einem geeigneten Passwort sichern:
  CODESYS-Menü [Projekt] > [Passwörter für Arbeitsgruppe...]

> Passwörter für ALLE Arbeitsgruppen festlegen!
> Arbeitsgruppen ohne Passwort haben ungehinderten Zugriff auf das Projekt.

Funktionsbausteine: zulässig für sicherheitsrelevante Funktionen

13425

Nur die folgenden FBs und Funktionen dürfen innerhalb der Sicherheitsfunktion oder zur Verarbeitung von sicherheitsrelevanten Daten verwendet werden:

- alle Funktionsbausteine der CODESYS-Bibliothek `Standard.LIB`
- alle CODESYS-Standard-Operatoren und Standard-Konvertierungen
- die CODESYS-Standard-Funktion `TIME()`
- alle Funktionsbausteine der ifm-Bibliothek `ifm_SafetyPLCopen_Vxxyyzz.LIB`
- die folgenden Funktionsbausteine der ifm-Bibliothek `ifm_CR7132_Vxxyyzz.LIB`:

CAN_SAFETY_RECEIVE (→ Seite 214)	empfängt eine sichere CAN-Nachricht (SRDO)
CAN_SAFETY_TRANSMIT (→ Seite 217)	überträgt eine sichere CAN-Nachricht (SRDO)
CHECK_DATA (→ Seite 375)	erzeugt über einen konfigurierbaren Speicherbereich eine Prüfsumme (CRC) und prüft die Daten des Speicherbereichs auf ungewollte Veränderung
ERROR_REPORT (→ Seite 382)	meldet dem System einen anwendungsspezifischen Fehler
ERROR_RESET (→ Seite 384)	setzt anstehende Fehlermeldungen zurück
FREQUENCY (→ Seite 300)	misst die Frequenz des am gewählten Kanal ankommenden Signals
FREQUENCY_PERIOD (→ Seite 302)	misst die Frequenz und die Periodendauer (Zykluszeit) in [µs] am angegebenen Kanal
INPUT_ANALOG (→ Seite 261)	Strom- und Spannungsmessung am analogen Eingangskanal
PACK_ERRORCODE (→ Seite 388)	hilft beim Zusammenbauen eines ERRORCODE aus den Bytes für: • Fehlerklasse • anwendungsspezifischer Fehler • Fehlerquelle • Fehlerursache
PERIOD (→ Seite 306)	misst am angegebenen Kanal die Frequenz und die Periodendauer (Zykluszeit) in [µs]
PERIOD_RATIO (→ Seite 308)	misst die Frequenz und die Periodendauer (Zykluszeit) in [µs] über die angegebenen Perioden am angegebenen Kanal. Zusätzlich wird das Puls-/Periodenverhältnis in [‰] angegeben.
SAFETY_SWITCH (→ Seite 268)	Betrieb der 1-kanaligen SafetySwitch der **ifm electronic gmbh**
SET_DEBUG (→ Seite 378)	organisiert (abhängig vom TEST-Eingang) den DEBUG-Modus oder den Monitoring-Modus
SET_IDENTITY (→ Seite 379)	setzt eine anwendungsspezifische Programmkennung
SET_INPUT_MODE (→ Seite 264)	weist einem Eingangskanal eine Betriebsart zu
SET_KEEP_ALIVE (→ Seite 386)	konfiguriert, welcher Ausgangskanal und welcher CANsafety-Kanal beim Auftreten eines bestimmten schweren Fehlers weiterbetrieben werden sollen, da sie von dem mit ERRORCODE gemeldeten Fehler unabhängig sind
SET_OUTPUT_MODE (→ Seite 313)	setzt die Betriebsart des gewählten Ausgangskanals
SHOW_ERROR_LIST (→ Seite 389)	liest den aktuell vorliegenden Fehler-Code
UNPACK_ERRORCODE (→ Seite 390)	hilft beim Trennen eines ERRORCODE in die Bytes für: • Fehlerklasse • anwendungsspezifischer Fehler • Fehlerquelle • Fehlerursache

Funktionsbausteine: zulässig im Anwendungsprogramm für nicht-sicherheitsrelevante Daten

14355

Die folgenden FBs und Funktionen dürfen innerhalb der Anwendung zur Verarbeitung von nicht-sicherheitsrelevanten Daten verwendet werden:

- alle Funktionsbausteine der CODESYS-Bibliothek `Standard.LIB`
- alle CODESYS-Standard-Operatoren und Standard-Konvertierungen
- die CODESYS-Standard-Funktion `TIME()`
- alle Funktionsbausteine der CODESYS-Bibliothek `SysLibStr.LIB`
- alle Funktionsbausteine der CODESYS-Bibliothek `Util.LIB`
- alle Funktionsbausteine der ifm-Bibliothek `ifm_SafetyPLCopen_Vxxyyzz.LIB`
- alle Funktionsbausteine der ifm-Bibliothek `ifm_CR7132_Vxxyyzz.LIB`
- alle Funktionsbausteine der ifm-Bibliothek `ifm_CR7132_CANopenxMaster_Vxxyyzz.LIB`
- alle Funktionsbausteine der ifm-Bibliothek `ifm_CR7132_CANopenxSlave_Vxxyyzz.LIB`
- alle Funktionsbausteine der ifm-Bibliothek `ifm_CR7132_J1939_Vxxyyzz.LIB`
- alle Funktionsbausteine der ifm-Bibliothek `ifm_hydraulic_32bit_Vxxyyzz.LIB`

Funktionsbausteine: Einschränkungen bei mehreren Instanzen

14877

Von den folgenden FBs sollte jeweils nur EINE Instanz im Anwendungsprogramm vorhanden sein. Ansonsten ist eine korrekte Ausführung der FBs gefährdet.

Funktionsbaustein
CANx
CANx_BAUDRATE
CANx_BUSLOAD
CANx_DOWNLOAD_ID
J1939_x
MEMORY_RETAIN_PARAM
SERIAL_SETUP
SET_PASSWORD

(x = 1...4 = Nummer der CAN-Schnittstelle)

Bei den folgenden FBs dürfen keine zwei oder mehr Instanzen im Anwendungsprogramm vorhanden sein, bei denen an den aufgeführten Eingängen der gleiche Wert übergeben wird. Ansonsten ist eine korrekte Ausführung der FBs nicht möglich.
Übergreifend über alle hiernach aufgeführten FBs darf keine Konfiguration vorgenommen werden, bei der mehr als eine Instanz die gleiche ID empfangen soll.

Funktionsbaustein	kritische FB-Eingänge
CANx_RECEIVE	ID
CANx_SDO_READ	NODE
CANx_SDO_WRITE	NODE
CAN_SAFETY_RECEIVE	NUMBER oder ID1 oder ID2
CAN_SAFETY_TRANSMIT	NUMBER oder ID1 oder ID2
J1939_x_GLOBAL_REQUEST	PG und PF und PS
J1939_x_RECEIVE	PG und PF und PS
J1939_x_RESPONSE	PG und PF und PS
J1939_x_SPECIFIC_REQUEST	DA und PG und PF und PS

(x = 1...4 = Nummer der CAN-Schnittstelle)

Bei den folgenden FBs dürfen keine zwei oder mehr Instanzen in einer Anwendung vorhanden sein, bei denen an den aufgeführten Eingängen der gleiche Wert übergeben wird. Ansonsten ist eine korrekte Ausführung der FBs nicht möglich.
Übergreifend über alle hiernach aufgeführten FBs darf keine Konfiguration vorgenommen werden, bei der mehr als eine Instanz auf denselben Eingangskanal zugreift.
Eine Ausnahme gibt es bei den FBs INPUT_ANALOG und SET_INPUT_MODE: hier darf jeweils eine Instanz auf denselben Eingangskanal zugreifen, wenn bei beiden Instanzen die selbe Einstellung bei MODE erfolgt.

Funktionsbaustein *)	kritische FB-Eingänge
FAST_COUNT(_E)	CHANNEL
FREQUENCY(_E)	CHANNEL
FREQUENCY_PERIOD(_E)	CHANNEL
INC_ENCODER(_E)	CHANNEL
INPUT_ANALOG(_E)	CHANNEL
PERIOD(_E)	CHANNEL
PERIOD_RATIO(_E)	CHANNEL
PHASE(_E)	CHANNEL
SET_INPUT_MODE(_E)	CHANNEL

*) "_E" nur bei den FBs für die Extended-Seite des ExtendedSafetyController

Bei den folgenden FBs dürfen keine zwei oder mehr Instanzen in einer Anwendung vorhanden sein, bei denen an den aufgeführten Eingängen der gleiche Wert übergeben wird. Ansonsten ist eine korrekte Ausführung der FBs nicht möglich.
Übergreifend über alle hiernach aufgeführten FBs darf keine Konfiguration vorgenommen werden, bei der mehr als eine Instanz auf denselben Ausgangskanal zugreift.
Diese Einschränkung gilt übergreifend über alle aufgeführten FBs, jedoch beim SET_OUTPUT_MODE nur, wenn dessen Eingang SAFETY den Wert TRUE hat.

Funktionsbaustein *)	kritische FB-Eingänge
OUTPUT_BRIDGE(_E)	CHANNEL
OUTPUT_CURRENT_CONTROL(_E)	OUTPUT_CHANNEL
PWM1000(_E)	CHANNEL
SAFETY_SWITCH	INPUT_CHANNEL
SET_OUTPUT_MODE(_E)	CHANNEL

*) "_E" nur bei den FBs für die Extended-Seite des ExtendedSafetyController

Bei den folgenden FBs dürfen keine zwei oder mehr Instanzen in einer Anwendung vorhanden sein, bei denen an den aufgeführten Eingängen der gleiche Wert übergeben wird. Ansonsten ist eine korrekte Ausführung der FBs nicht möglich.

Funktionsbaustein *)	kritische FB-Eingänge
OUTPUT_CURRENT(_E)	OUTPUT_CHANNEL

*) "_E" nur bei den FBs für die Extended-Seite des ExtendedSafetyController

Funktionsbausteine: nicht zulässig im Anwendungsprogramm

15436

► Die nachfolgend aufgeführten Funktionsbausteine im Anwendungsprogramm NICHT aufrufen!

- Die Funktionen der externen CODESYS-Bibliothek "SysLibFile.LIB" werden im Laufzeitsystem ausgeführt. Sie werden von CODESYS und dem Service-Tool über die Maintenance-Schnittstellen verwendet.
  Diese Funktionen im Anwendungsprogramm NICHT aufrufen!

Bibliotheken: vom System für CANopen erforderlich

14356

Die folgenden Bibliotheken werden bei Verwendung der CANopen-Funktionalität automatisch in das CODESYS-Projekt eingebunden:

- die CODESYS-Bibliothek `3S_CanDrvOptTableEx.LIB`
- die CODESYS-Bibliothek `3S_CANopenMasterOptTableEx.LIB`
- die CODESYS-Bibliothek `3S_CANopenManagerOptTableEx.LIB`
- die CODESYS-Bibliothek `3S_CanOpenDeviceOptTableEx.LIB`
- die CODESYS-Bibliothek `3S_CanOpenNetVarOptTableEx.LIB`
- die CODESYS-Bibliothek `SysLibCallback.LIB`

> ! Die darin enthaltenen Funktionsbausteine und Funktionen dürfen NICHT im Code des Anwendungsprogramms direkt aufgerufen werden!

3.5.7 Regel 7 – Verwendung von Variablen

13304

- Für Variablen nur symbolische Namen verwenden.
  Die IEC-Adressen werden nur in der Variablendeklaration eingesetzt.
  Das gilt auch für die IEC-Adressen von Merkern (%M...).
- Für jede Variable eine eigene Deklarationszeile verwenden.
  Keine Aufzählung gleicher Variablentypen in gemeinsamer Deklarationszeile!

Gutes Beispiel:	Schlechtes Beispiel:
VAR A: BOOL; (* 1st Variable *) B: BOOL; (* 2nd Variable *) C: BOOL; (* 3rd Variable *) END_VAR	VAR A, B, C: BOOL; (* Some Variables *) END_VAR

- Die Funktion der Variablen zur deutlichen Unterscheidung mit einer dem Variablennamen vorangestellten Buchstabenkombination kennzeichnen:

S_	sicherheitsrelevantes Element
G_	globales Element
GS_	sicherheitsrelevantes globales Element
I_	Eingangsvariable
O_	Ausgangsvariable

Beispiel:

```
VAR_INPUT
   I_VARIABLE: BYTE; (* Eingangs-Variable *)
   I_...
END_VAR
VAR_OUTPUT
   O_VARIABLE: WORD; (* Ausgangs-Variable *)
   O_...
END_VAR
```

- Jede Variable mit einem eindeutigen, selbstsprechenden und aussagekräftigen Namen bezeichnen.
  Diese Variablennamen auch in den Kommentaren des Quelltextes nutzen.
- Werte von Variablen auf Plausibilität prüfen:
  Wertevergleich nie auf "gleich" (=), sondern auf "größergleich" (≥) oder "kleinergleich" (≤), weil möglicherweise niemals genau "gleich" erreicht oder im Messzyklus gesehen wird.
- Die Gültigkeit von Variablenwerten sofort innerhalb eines Funktionsbausteins überprüfen und sofort aufdecken.
  Beispiel: einen Wert durch einen Vergleichstest kontrollieren, um die Einhaltung des Wertebereichs sicherzustellen
  Eine Variablenprüfung außerhalb des Funktionsbausteins führt zu einer unklaren Programmstruktur.
- Den Programmablauf NICHT vom Wert einer Variablen abhängig machen, die während der Laufzeit berechnet wird. Andernfalls besteht die Gefahr von zufälligen Fehlern im Programmablauf.
  Jedoch logische Sprünge und Programmverzweigungen sind zulässig.
- Sichere Funktionsbausteine sollen Werte von globalen Variablen nicht verändern.
- Nicht-sichere Bausteine dürfen nicht auf sichere Variablen schreibend zugreifen. Dies in CODESYS mit der Funktion [Projekt] > [Querverweisliste ausgeben] überprüfen.
- Sichere Bausteine dürfen nicht auf nicht-sichere Variablen lesend zugreifen und die Werte für die Sicherheitsfunktion verwenden. Dies in CODESYS mit der Funktion [Projekt] > [Querverweisliste ausgeben] überprüfen.

- ► Jeden Ausgang und jede Variable an nur EINER Stelle im Programm einschalten (setzen) und nur an EINER Stelle ausschalten (rücksetzen): zentralisierte Bedingungen.
- ► Datenstrukturen und Daten-Arrays, die zum Lesen oder Schreiben mehrere Zyklen erfordern, während des Schreib-/Lesezyklus vor Veränderung schützen.
- ► Adressen möglichst NICHT mehrfach verwenden, wegen schwer durchschaubarer Nebeneffekte. Soll auf eine Word-Variable wort- und bitweise zugegriffen werden, dann eine Variable für das Wort definieren und bitweise mit Hilfe des Bit-Zugriffs `Variable.Bitnummer` zugreifen. Beispiele:

Gutes Beispiel:	Schlechtes Beispiel:

`VAR CONSTANT`
`  EnableBit: INT:=0;`
`END_VAR`
`VAR_GLOBAL`
`  Flags AT %QW12: WORD;`
`END_VAR`
 	
`Flags:=0;`
`Flags.0:=TRUE;`
	`VAR_GLOBAL`
	`  Flags AT %QW12: WORD;`
	`  Enable AT %QX12.0: BOOL;`
	`END_VAR`
	
	`Flags:=0;`
	`Enable:=TRUE;`

- ► Für sichere Retain-Variable explizite Testfälle für den Powerdown-Fall vorsehen.

Variablen: zulässig für sicherheitsrelevante Daten

13426

Nur die folgenden Variablen dürfen zur Verarbeitung von sicherheitsrelevanten Daten verwendet werden:

Systemmerker (Symbolname)	Typ	Beschreibung
ANALOGxx xx = 00...15	WORD	Analog-Eingang xx: gefilterter A/D-Wandler-Rohwert (12 Bit) ohne Kalibrierung und Normierung
CAN_SAFETY_ERROR_1	BOOL	Fehler bei den CANsafety-Nachrichten an CANsafety-Kanal 1 TRUE: schwerer Fehler aufgetreten FALSE: kein Fehler
CAN_SAFETY_ERROR_2	BOOL	Fehler bei den CANsafety-Nachrichten an CANsafety-Kanal 2 TRUE: schwerer Fehler aufgetreten FALSE: kein Fehler
CLAMP_15_VOLTAGE	WORD	Spannung an Klemme 15 in [mV]
ERROR	BOOL	TRUE: sicherer Zustand eingenommen FALSE: kein schwerer Fehler aufgetreten
ERROR_BREAK_Ix (x=0...n; Wert abhängig vom Gerät, → Datenblatt)	DWORD	Leiterbruch-Fehler an der Eingangsgruppe x Wenn Eingang = SAFETY ⇨ schwerer Fehler! [Bit 0 für Eingang 0] ... [Bit z für Eingang z] dieser Gruppe Bit = TRUE: Fehler Bit = FALSE: kein Fehler
ERROR_BREAK_Qx (x=0...n; Wert abhängig vom Gerät, → Datenblatt)	DWORD	Leiterbruch-Fehler an der Ausgangsgruppe x Wenn Ausgang = SAFETY ⇨ schwerer Fehler! [Bit 0 für Ausgang 0] ... [Bit z für Ausgang z] dieser Gruppe Bit = TRUE: Fehler Bit = FALSE: kein Fehler
ERROR_CURRENT_Ix (x=0...n; Wert abhängig vom Gerät, → Datenblatt)	DWORD	Überstrom-Fehler an der Eingangsgruppe x [Bit 0 für Eingang 0] ... [Bit z für Eingang z] dieser Gruppe Bit = TRUE: Fehler Bit = FALSE: kein Fehler
ERROR_OVERLOAD_Qx (x=0...n; Wert abhängig vom Gerät, → Datenblatt)	DWORD	Überlast-Fehler an der Ausgangsgruppe x [Bit 0 für Ausgang 0] ... [Bit z für Ausgang z] dieser Gruppe Bit = TRUE: Fehler Bit = FALSE: kein Fehler
ERROR_POWER	BOOL	Spannungs-Fehler für SUPPLY_VOLTAGE: TRUE: Wert außerhalb des zulässigen Bereichs > schwerer Fehler FALSE: Wert in Ordnung
ERROR_SAFETY_Ix (x=0...n; Wert abhängig vom Gerät, → Datenblatt)	DWORD	schwerer Fehler an der sicheren Eingangsgruppe x [Bit 0 für Eingang 0] ... [Bit z für Eingang z] dieser Gruppe Bit = TRUE: Fehler Bit = FALSE: kein Fehler
ERROR_SAFETY_Qx (x=0...n; Wert abhängig vom Gerät, → Datenblatt)	DWORD	schwererFehler an der sicheren Ausgangsgruppe x [Bit 0 für Ausgang 0] ... [Bit z für Ausgang z] dieser Gruppe Bit = TRUE: Fehler Bit = FALSE: kein Fehler
ERROR_SHORT_Ix (x=0...n; Wert abhängig vom Gerät, → Datenblatt)	DWORD	Kurzschluss-Fehler an der Eingangsgruppe x Wenn Eingang = SAFETY ⇨ schwerer Fehler! [Bit 0 für Eingang 0] ... [Bit z für Eingang z] dieser Gruppe Bit = TRUE: Fehler Bit = FALSE: kein Fehler

Systemmerker (Symbolname)	Typ	Beschreibung
ERROR_SHORT_Qx (x=0...n; Wert abhängig vom Gerät, → Datenblatt)	DWORD	Kurzschluss-Fehler an der Ausgangsgruppe x Wenn Ausgang = SAFETY ⇨ schwerer Fehler! [Bit 0 für Ausgang 0] ... [Bit z für Ausgang z] dieser Gruppe Bit = TRUE: Fehler Bit = FALSE: kein Fehler
ERROR_SYSTEM	BOOL	System-Fehler (nur sichtbar im TEST-Betrieb) TRUE: Geräte-Hardware defekt > Fataler Fehler ▶ Gerät an ifm senden FALSE: kein Fehler
ERROR_TEMPERATURE	BOOL	Temperatur-Fehler TRUE: Wert außerhalb des zulässigen Bereichs > fataler Fehler FALSE: Wert in Ordnung
ERROR_VBBx	BOOL	Versorgungsspannungs-Fehler an VBBx (x = O | R): TRUE: Wert außerhalb des zulässigen Bereichs > schwerer Fehler FALSE: Wert in Ordnung
ERRORCODE	DWORD	Zuletzt eingetragener Fehler in der internen Fehlerliste Die Liste enthält alle aufgetretenen Fehler-Codes.
Ixx xx = 00...15	BOOL	Status am Binäreingang xx Voraussetzung: Eingang ist als Binäreingang konfiguriert (MODE = IN_DIGITAL_H oder IN_DIGITAL_L) TRUE: Spannung am Binäreingang > 70 % von VBBS FALSE: Spannung am Binäreingang < 30 % von VBBS oder: nicht als Binäreingang konfiguriert oder: falsch konfiguriert
Qxx xx = 00...15	BOOL	Status am Binärausgang xx: Voraussetzung: Ausgang ist als Binärausgang konfiguriert TRUE: Ausgang aktiviert FALSE: Ausgang deaktiviert (= Initialwert) oder: nicht als Binärausgang konfiguriert
RELAIS_VBBy y = O | R	BOOL	TRUE: Relais für VBBy aktiviert Ausgangsgruppe x wird mit Spannung versorgt (x = 1 | 2) FALSE: Relais für VBBy ausgeschaltet Ausgangsgruppe x ist spannungslos
SUPPLY_VOLTAGE	WORD	Versorgungsspannung an VBBs in [mV]
TEST	BOOL	TRUE: Test-Eingang ist aktiv: • Programmiermodus ist freigeben • Software-Download ist möglich • die sicheren Ausgänge sind deaktiviert • es werden keine CANsafety-Nachrichten versendet • Zustand des Anwendungsprogramms ist abfragbar • kein Schutz der gespeicherten Software möglich FALSE: laufender Betrieb der Anwendung
VBBx_RELAIS_VOLTAGE x = O | R	WORD	Versorgungsspannung an VBBx nach Relaiskontakt in [mV]
VBBx_VOLTAGE x = O | R	WORD	Versorgungsspannung an VBBx in [mV]

3.5.8 Regel 8 – Verwenden von Datentypen

13305

Von den in CODESYS definierten Datentypen sind folgende in sicherheitsrelevanten Anwendungen zugelassen:

Datentyp	zulässig (empfohlen) für sicherheitsrelevante Anwendungen	
BOOL	ja	
BYTE SINT USINT	ja	
WORD INT UINT	ja	
DWORD DINT UDINT	ja	
TIME TOD DATE DT	ja	
STRING	bedingt	Einsatz wegen fehlender sicherer Ein-/Ausgabegeräte wenig sinnvoll
LREAL REAL	bedingt	Fehleranfällig durch Rundungsfehler, daher keine Abfrage mit EQ-Operator möglich. Ungültige Operationen beachten!
ARRAY	bedingt	Nur mit expliziter Bereichsüberprüfung!
STRUCT	ja	
Aufzählungs-Typen	ja	
Unterbereichs-Typen	ja	
POINTER	bedingt	Keine Pointer-Arithmetik! Bereichsüberprüfung! Neue Zuweisung des Pointerwertes zu Beginn von jedem Zyklus!

► Bei einer Bereichsüber- oder Bereichsunterschreitung, die nicht durch die Anwendung erklärbar ist, die Steuerung in den sicheren Zustand bringen!

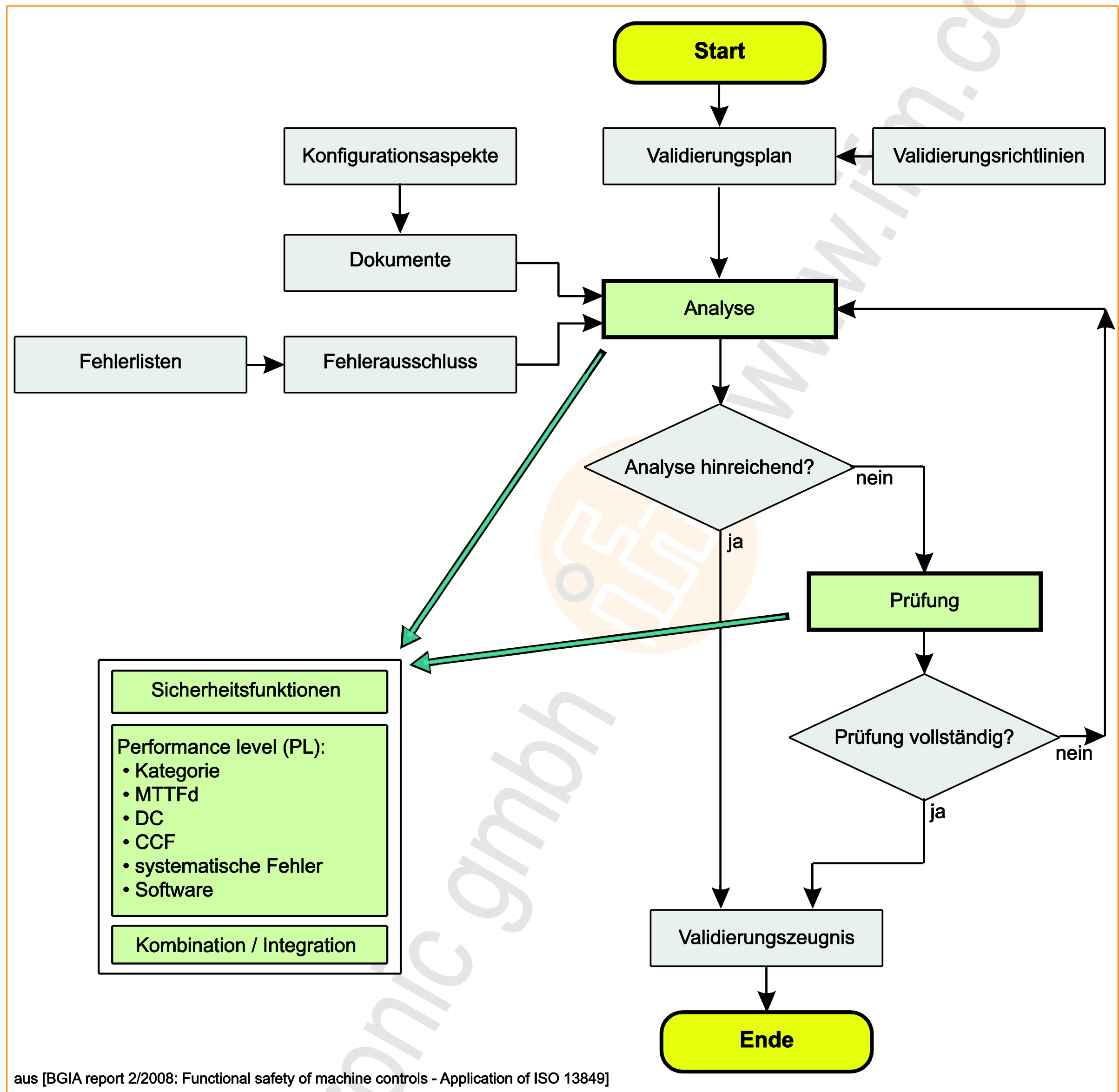
3.5.9 Regel 9 – Testen und Handling sicherheitsrelevanter Software

Inhalt

13306

Übersicht: Verifikation und Validierung nach ISO 13849-2

14172



aus [BGIA report 2/2008: Functional safety of machine controls - Application of ISO 13849]

Legende:

- Validation guidelines = Validierungsleitsätze:
  abgeleitet aus der ISO 13849-1
- Validation plan = Validierungsplan:
  enthält verbindlich festgelegt alle geplanten Aktivitäten
- Configuration aspects = Gestaltungsaspekte:
  • Anforderungen an Spezifikation und Dokumentation,
  • notwendige Maßnahmen zur Beherrschung systematischer Ausfälle,
  • ergonomische Gestaltungsaspekte,
  • Kategorien,
  • Anforderungen an Software
- Documents = Dokumente:
  Dokumente, die im Rahmen der Entwicklung entstanden sind
- Error lists = Fehlerlisten:
  abgeleitet aus den Anhängen zur ISO 13849-2
- Analysis = Analyse:
  Beurteilung, ob die spezifizierten Anforderungen erfüllt wurden

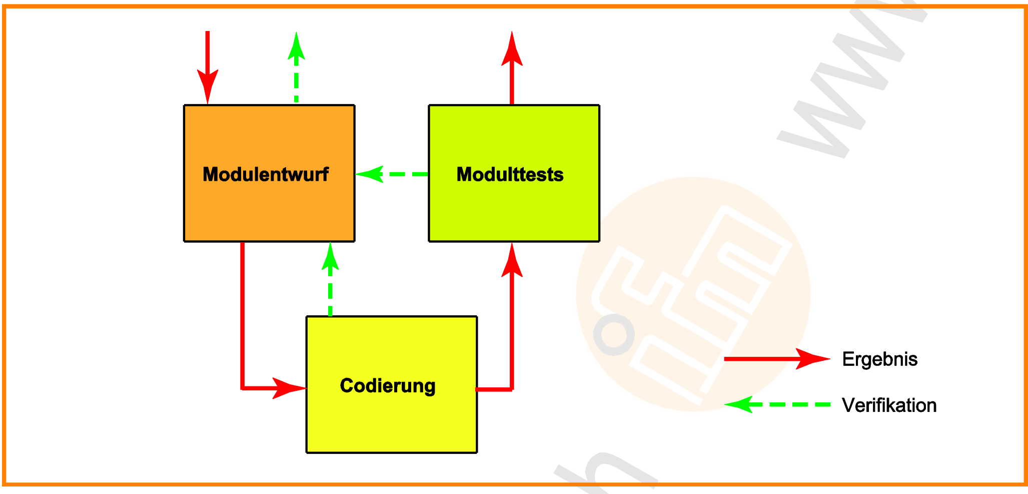
- Verification = Prüfung:
  Wenn Begutachtung durch Analyse nicht ausreicht oder nicht möglich ist, dann Sicherheitsfunktion prüfen!
- Verification report = Validierungsbericht:
  Analysen und Prüfungen inklusive ihrer Ergebnisse dokumentieren!

Modultest

13308

► Um Programmierfehler möglichst frühzeitig zu entdecken, sollten die einzelnen Programmmodule (→ Regel 6) parallel zu Entwicklung immer wieder getestet werden. Diese Modultests können die einzelne Funktion isoliert prüfen und die Wirksamkeit der Sicherheitsfunktion nachweisen.

► Diese Tests nach einem festgelegten Testplan und in einer festgelegten Testreihenfolge auf Basis des Modulentwurfs durchführen.



Grafik: Der Modultest im V-Modell

Beim Modultest kann – je nach Anforderung – der Test-Eingang des SafetyControllers aktiviert sein oder nicht aktiviert sein.

(!) Bei aktiviertem Test-Eingang sind bestimmte Sicherheitsfunktionen nicht oder nur teilweise aktiv (→ folgendes Kapitel).

► Den Testablauf und die Ergebnisse (auch gefundene Fehler, die beseitigt wurden) in der Projektdokumentation festhalten.

Verhalten der sicherheitrelevanten Ausgänge im MONITORING- und DEBUG-Modus

13307

Controller-Eingang TEST	FB SET_DEBUG Eingang DEBUG	Variablenwerte	als "SAFETY" konfigurierte Ausgänge
nicht mit VBB verbunden	TRUE	MONITORING-Modus: nur lesen möglich	aktiv
mit VBB verbunden	TRUE oder FALSE	DEBUG-Modus: lesen und ändern möglich	deaktiviert

Tabelle: Verhalten der sicherheitsrelevanten Ausgänge im MONITORING- und DEBUG-Modus

HINWEIS

Der Debug-Modus kann bestehen bleiben, auch wenn das Anwendungsprogramm ohne erneutes Verwenden von SET_DEBUG aktualisiert wurde.
> Ein fortgesetzter Lesezugriff kann möglich sein.
> Ein fortgesetzter Schreibzugriff ist nicht mehr möglich.

▶ Nach Aktualisieren des Anwendungsprogramms ein Power-On-Reset durchführen!
Somit wird der Debug-Modus zuverlässig unterbrochen.

Details:

SET_DEBUG (→ Seite 378)	organisiert (abhängig vom TEST-Eingang) den DEBUG-Modus oder den Monitoring-Modus

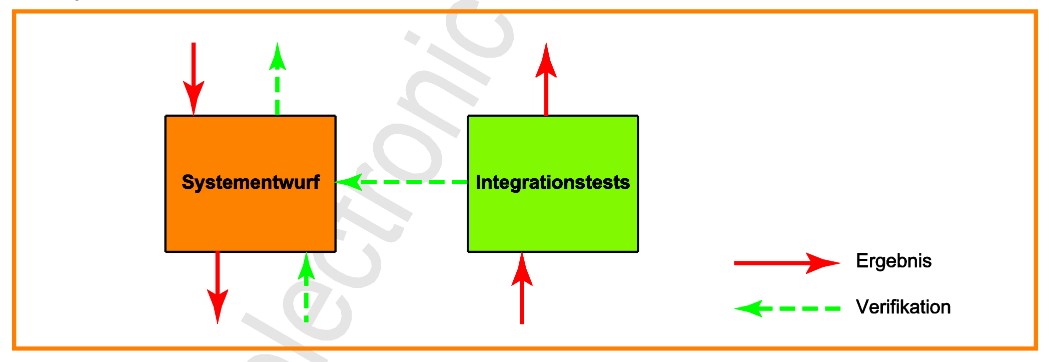
Test-Eingang → Montageanleitung > Kapitel "Technische Daten" > Kapitel "Anschlussbelegung"

Integrationstest

13309

Nachdem alle einzelnen Module getestet sind, erfolgt als nächster Schritt ein Integrationstest in der mobilen Maschine.

Der Integrationstest betrachtet nur das Zusammenspiel der einzelnen Funktionsmodule, so wie es im Systemdesign festgelegt wurde. Die I/O-Tests müssen sicherstellen, dass die sicheren Signale des SafetyControllers korrekt verwendet werden.



Grafik: Der Integrationstest im V-Modell

Beim Integrationstest darf der Test-Eingang des SafetyControllers nicht aktiviert sein.

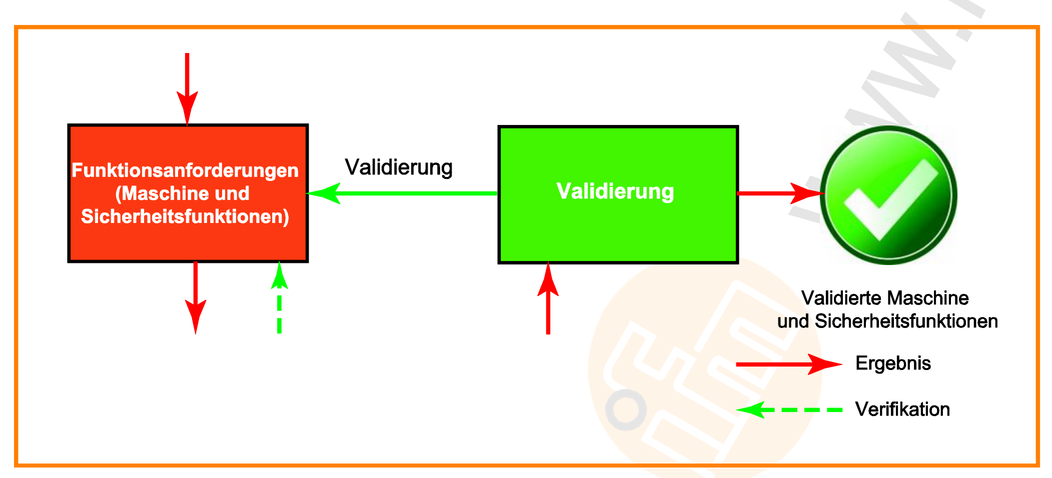
Validierung

12261

Nach dem Abschluss aller vorherigen Tests:

► Als letzter Schritt muss die Validierung der gesamten Maschine und des Anwendungsprogramms gegen das Pflichtenheft und das Maschinenkonzept erfolgen.

! Bei der Validierung darf der TEST-Eingang des SafetyControllers NICHT aktiviert sein.



Grafik: Die Validierung im V-Modell

► Dabei final Folgendes überprüfen:
  • Sicherheitsfunktionen und das Sicherheitskonzept
  • die damit verbundenen Kennwerte (PL und Kategorie)

► Zum Nachweis praxistaugliche Fehlersimulationen an den sicheren Ein- und Ausgängen durchführen, z.B. Leiterbruch oder Querschluss.

► Erst nach diesem Schritt das Anwendungsprogramm zur Vervielfältigung aus dem Gerät auslesen und archivieren.

► Anschließend das ausgelesene Programm wie nachfolgend beschrieben vervielfältigen.

Handhabung von sicherheitsrelevanter Software

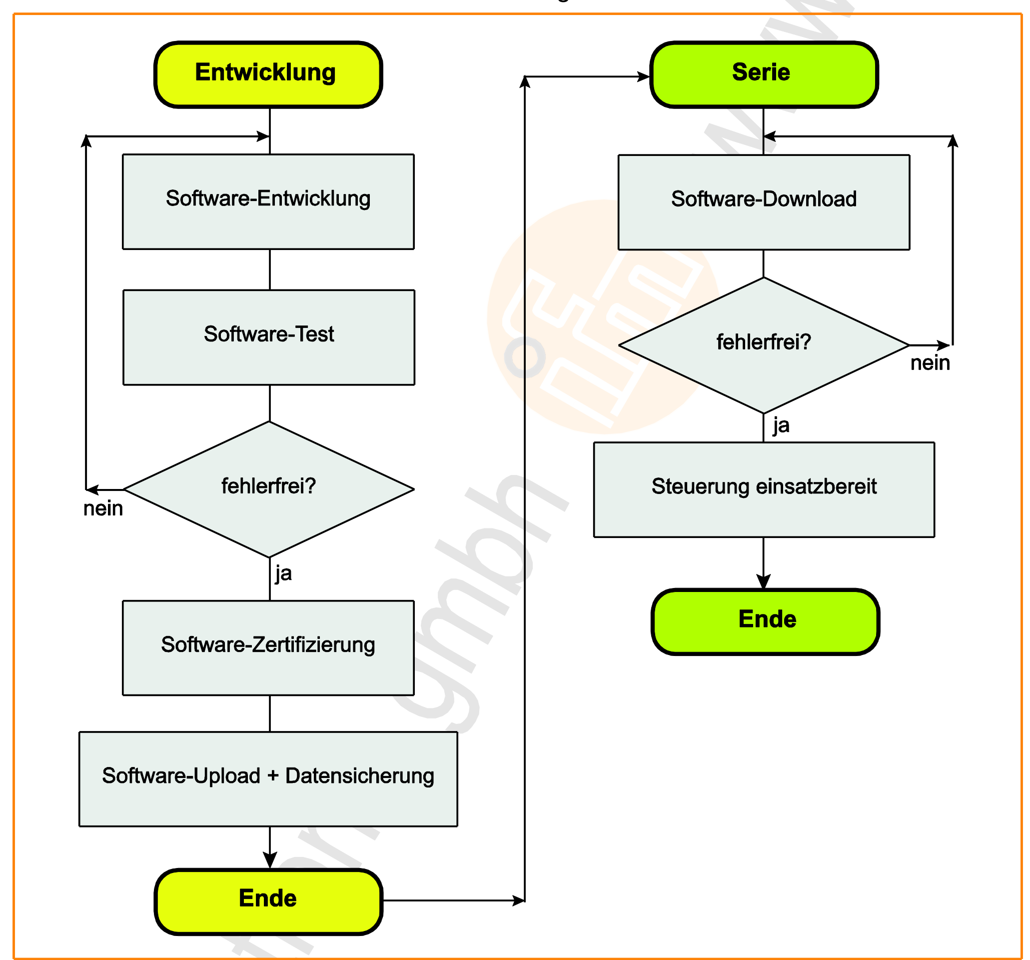
13312

Das Anwendungsprogramm wird mit dem Programmiersystem CODESYS erstellt und während der Programmentwicklung mehrfach zum Testen in die Steuerung geladen:

In CODESYS: [Online] > [Projekt in Steuerung laden].

Für jeden derartigen Download via CODESYS wird dazu der Quellcode neu übersetzt. Daraus resultiert, dass auch jedes Mal im Speicher der Steuerung eine neue Prüfsumme gebildet wird. Auch für Sicherheitssteuerungen ist dieses Verfahren bis zur Freigabe der Software zulässig.

Zumindest für sicherheitsrelevante Anwendungen muss aber für die Serienproduktion der Maschine eine Einheitlichkeit der Software und ihrer Prüfsumme gewährleistet sein.



Grafik: Erstellen und Verteilen von zertifizierter Software
Upload = Daten / Programm vom Controller in den PC laden
Download = Daten / Programm vom PC in den Controller laden

Sichern der freigegebenen Software

13314

Nach Abschluss des Integrationstests und Freigabe des Gesamtsystems:

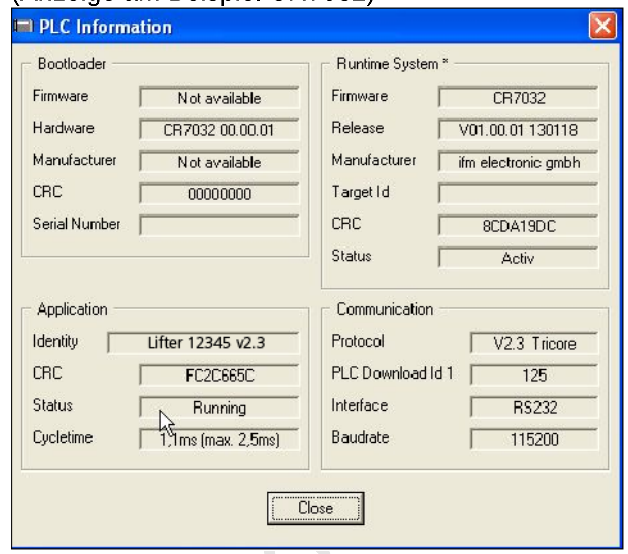
► Die letzte Version des Anwendungsprogramms aus der Steuerung auslesen.
Das dazu geeignete Werkzeug*) ist:
• **ifm**-Downloader

> **HINWEIS**
>
> ► Sicherheitsrelevante Daten NICHT per **ifm**-Downloader aus dem Gerät auslesen oder in Geräte laden!
> Hierfür sind keine geeigneten Sicherungsmaßnahmen vorhanden.
>
> • Sicherheitsrelevante Programme hingegen dürfen mit dem **ifm**-Downloader gelesen und kopiert werden.

► Datei auf einem Datenträger speichern als `name_der_projektdatei.H86`.

Nur dieses Verfahren gewährleistet Folgendes für das Anwendungsprogramm:
• es ist mit den entsprechenden Prüfsummen gesichert
• es ist unverfälscht gespeichert
• es wird unverfälscht in das Seriengerät geladen

► Mit dem Downloader die Prüfsumme (CRC) und die Identity des Anwendungsprogramms auslesen:
• Im Downloader-Menü: [Specials] > [PLC Information]
(Anzeige am Beispiel CR7032)



• Screenshot davon archivieren.

*) Dieses Werkzeug dient dem einfachen Übertragen des Programmcodes vom Programmierplatz in die Steuerung. Grundsätzlich kann jedes Anwendungsprogramm mit dem Werkzeug auf die Steuerungen kopiert werden. Sicherheitsrelevante Anwendungsprogramm MÜSSEN mit diesem Werkzeug auf die Steuerungen kopiert werden, um die Prüfsumme CRC (mit der die Software zertifiziert wurde) nicht zu verfälschen.

3.5.10 Regel 10 – Zertifizierung

13315

Sollte oder muss eine unabhängige Organisation das Maschinenkonzept und die Entwicklung der Sicherheitsfunktionen einer mobilen Maschine zertifizieren?

- Dies ist immer vom Einzelfall abhängig.
- Dies müssen der Maschinenbauer und der Programmierer in eigener Verantwortung entscheiden.
- In einigen Fällen ergibt sich das richtige Vorgehen auch aus der Produktnorm (C-Norm).

Grundsätzlich gibt es zwei mögliche Wege für eine Zertifizierung:

- die Zertifizierung durch eine unabhängige Organisation (z.B. TÜV, MIRA).
- die Selbst-Zertifizierung durch den Maschinenbauer.
  Gerade bei kleinen und wenig komplexen Anwendungen oder bei schon mehrfach entwickelten Maschinen lassen sich mit der Selbst-Zertifizierung der externe Aufwand und damit die Kosten minimieren.

Jedoch bleibt der folgende Aufwand in beiden Fällen gleich (Beispiele):
• Konzeptentwurf
• Systemdesign
• Dokumentation
• Tests

Auch bei der Selbst-Zertifizierung trägt der Maschinenbauer die komplette Verantwortung speziell für den Konzeptentwurf und das Systemdesign. Da hierbei das 4-Augen-Prinzip durch einen unabhängigen Prüfer fehlt, können im Schadensfall erhebliche rechtliche Konsequenzen auf das Unternehmen zukommen.

Daher empfiehlt sich in den meisten Fällen die Zertifizierung durch eine unabhängige Organisation. Diese prüft im ersten Schritt Folgendes:
• die geplante Entwicklungsstruktur,
• den Konzeptentwurf
• das Systemdesign.

Abschließend führt die unabhängige Organisation den Integrationstest durch.

Durch das Einbeziehen eines unabhängigen Prüfers resultieren folgende Vorteile:

- schwerwiegende Fehler beim Maschinenkonzept werden vermieden,
- die rechtlichen Konsequenzen im Schadensfall für das Unternehmen werden auf ein vertretbares Risiko minimiert.

3.5.11 Regel 11 – Inbetriebnahme und Wartung der Steuerung beim Zugriff über CAN

14308

► Vor dem Integrieren der Geräte in ein bestehendes Netzwerk die Download-IDs korrekt einstellen! Zum Einstellen das Gerät über RS232 oder CAN-Direktverbindung mit dem PC verbinden.

Beim Zugriff auf das Gerät über den CAN-Bus gelten die folgende Regeln:

► Bei allen Wartungsarbeiten über den CAN-Bus sicherstellen, dass der Zugriff auf die richtige Steuerung erfolgt. Der Zugriff erfolgt per SDOs des CANopen-Protokolls. Die SDO-Identifier für die Kommunikation zwischen Programmier- oder Service-Tool und der Steuerung werden anhand einer in der Steuerung gespeicherten Download-ID (Kennung für Programm-Download und Wartungszugriff) plus Basisadresse des SDOs ermittelt.
→ Kapitel ***IDs (Adressen) in CAN*** (→ Seite 157)
Diese Download-ID alleine reicht zum Identifizieren der Steuerung nicht aus, da die Nummer versehentlich geändert werden kann.

Es gibt 2 Methoden, um sicherzustellen, dass der Wartungszugriff über CAN auf die richtige Steuerung erfolgt:

► In der Fertigung des Anwenders einen Netzwerkplan mit allen Steuerungen in der Maschine erstellen, in den die Seriennummer jeder verbauten Sicherheitssteuerung eingetragen wird.
Vor dem Download einer Software-Komponente die Seriennummer auslesen (z.B. mit dem Downloader) und mit Hilfe des Netzwerkplans überprüfen, dass man auf die richtige Steuerung zugreift!

► Die TEST-Eingänge aller Sicherheitssteuerungen in der Maschine einzeln verdrahten und eindeutig markieren, so dass eine Zuordnung zu den Steuerungen eindeutig hergestellt werden kann!
Bei einem Wartungszugriff immer nur den TEST-Eingang der einen Steuerung aktivieren, auf die zugegriffen werden soll!

3.5.12 Regel 12 – Ablauf für sicherheitsrelevante Anwendungen in der Produktion

13316

ifm liefert die Geräte in einer mit einem Siegelaufkleber verschlossenen Verpackung aus. Nur wenn dieser Siegelaufkleber unbeschädigt ist, kann davon ausgegangen werden, dass das Gerät beim Kunden in dem Zustand angekommen ist, wie es die Fertigung beim Hersteller verlassen hat.

► Die nachfolgenden Punkte unbedingt beachten!

Nachverfolgung der verbauten Sicherheitssteuerungen

13317

Grundsätzlich sind alle Sicherheitssteuerungen mit einer eindeutigen Seriennummer versehen.

► Vor Einbau der Geräte diese Seriennummer zusammen mit der Maschinenkennung und dem Einsatzort der Maschine dokumentieren und archivieren!

Nur so können zu einem späteren Zeitpunkt – auch nach Jahren – gezielt einzelne Maschinen bei Fehlern überarbeitet und bei Bedarf Komponenten ausgetauscht werden.

Download der freigegebenen Software

13318

In der Serienproduktion von sicherheitsrelevanten Anwendungen müssen alle Maschinen dieser Serie eine einheitliche Software (Laufzeitsystem + Anwendungsprogramm) erhalten.

► Dazu nur die mit einer CRC gesicherte und gespeicherte Datei mit dem **ifm**-Downloader in die Steuerungen laden.

> Durch die integrierte Prüfsumme erkennt der **ifm**-Downloader beim erneuten Laden automatisch einen Fehler in den Daten dieser Datei.

> Der **ifm**-Downloader verhindert somit eine Fehlfunktion der Sicherheitssteuerung.

> Ein Download des Anwendungsprogramms mit den Programmiersystem CODESYS ist nicht zulässig!
>
> > Die Prüfsumme wäre verändert.
>
> > Das Anwendungsprogramm wäre nicht mehr eindeutig identifizierbar.

CODESYS-Projekt mit Passwort sichern

14169

► Den Zugriff von nicht-autorisierten Personen auf die Software verhindern!
Dazu das CODESYS-Projekt mit einem geeigneten Passwort sichern:
CODESYS-Menü [Projekt] > [Passwörter für Arbeitsgruppe...]

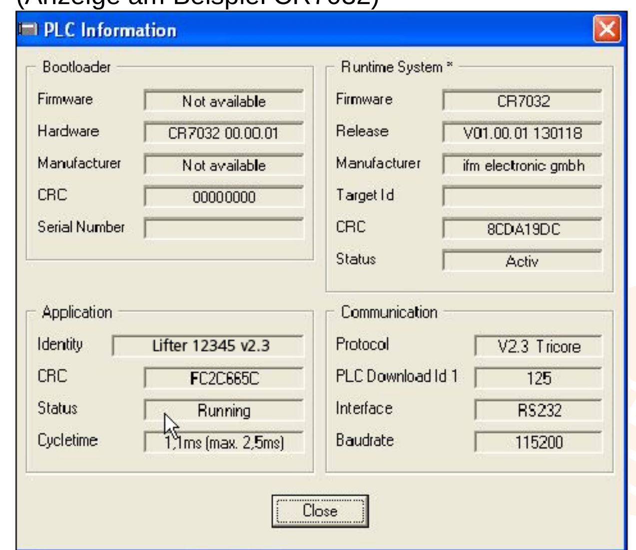
> Passwörter für ALLE Arbeitsgruppen festlegen!
> Arbeitsgruppen ohne Passwort haben ungehinderten Zugriff auf das Projekt.

Die übertragene Software prüfen

13789

► Nach dem Download mit dem **ifm**-Werkzeug prüfen:
  • wurde das richtige Laufzeitsystem (plus Version) auf das Gerät geladen?
  • wurde das richtige Anwendungsprogramm (plus Version) auf das Gerät geladen?

► Mit dem Downloader die Prüfsumme (CRC) des Anwendungsprogramms auslesen:
  • Im Downloader-Menü: [Specials] > [PLC Information]
(Anzeige am Beispiel CR7032)



  • Darstellung mit den Freigabe-Unterlagen vergleichen
  (→ Kapitel ***Sichern der freigegebenen Software*** (→ Seite 120)).

► Im Fehlerfall mit der korrekten Software den Download wiederholen.

3.5.13 Regel 13 – Nachträgliche Programmänderungen

13319

Die sicherheitsrelevante Software ist freigegeben und zertifiziert.
Nun sollten möglichst keine Änderungen mehr erfolgen.

Wurden dennoch nachträgliche Änderungen erforderlich?

► Vor der Änderung überprüfen, ob die Spezifikation des sicherheitsrelevanten Anwendungsprogramms verletzt wird:
  • durch eine neue Risikobetrachtung und
  • auf jeden Fall durch eine neue Einfluss-Analyse.

► Den geänderten Teil der Software erneut prüfen, dokumentieren und gegebenenfalls neu zertifizieren!

- Ob eine erneute Zertifizierung nötig ist, ist immer vom Einzelfall abhängig. Dies müssen der Maschinenbauer und der Programmierer in eigener Verantwortung entscheiden.
- Auf eine erneute Zertifizierung kann gegebenenfalls verzichtet werden, wenn KEINE sicherheitsrelevanten Elemente verändert, hinzugefügt oder entfernt werden.

► In der Dokumentation die Änderungshistorie festhalten:
  • warum wurde geändert?
  • wer hat geändert?
  • wann wurde geändert?
  • was wurde geändert?
  • die neue Einfuss-Analyse und
  • die neue Risikobetrachtung

(!) Eine nachträgliche Programmänderung, die von diesem Ablauf abweicht...
• ist nicht zulässig
• führt zum Verlust des Zertifikats
• und kann zudem zum Verlust der Sicherheitsfunktion führen.

4 Systembeschreibung

Inhalt

975

4.1 Angaben zum Gerät

13341

Diese Anleitung beschreibt aus der Gerätefamilie für den mobilen Einsatz, **ecomat*mobile*** der **ifm electronic gmbh**:

- ExtendedSafetyController: CR7132

Dieses Gerät eignet sich für den Einsatz in sicherheitsrelevanten Anwendungen:
• bis zu PL d nach ISO 13849,
• bis zu SIL CL 2 nach IEC 62061.

Die Ein- und Ausänge der Extended-Seite sind NICHT für sicherheitsrelevante Funktionen nutzbar.

4.2 Hardware-Beschreibung

Inhalt

14081

4.2.1 Hardware-Aufbau

12244

Das Gerät startet erst, wenn am Versorgungsanschluss VBBs (unter anderem Versorgung der Relais auf der Standardseite) und an Klemme 15 eine ausreichende Spannung anliegt.
Klemme 15 ist in Fahrzeugen die vom Zündschloss geschaltete Plusleitung.
Als ausreichende Spannung gilt > 10 V
Zulässige Betriebsspannung → Datenblatt

Der ExtendedController verfügt über 4 interne Ausgangsrelais:
- Standard-Seite: 2 Relais trennen jeweils 8 Ausgänge von der Klemmenspannung VBBx (x=O|R),
- Extended-Seite: 2 Relais trennen jeweils 16 Ausgänge von der Klemmenspannung VBBx (x=1|2|3|4).

Das Trennen erfolgt mit Ausschalten der Relais.

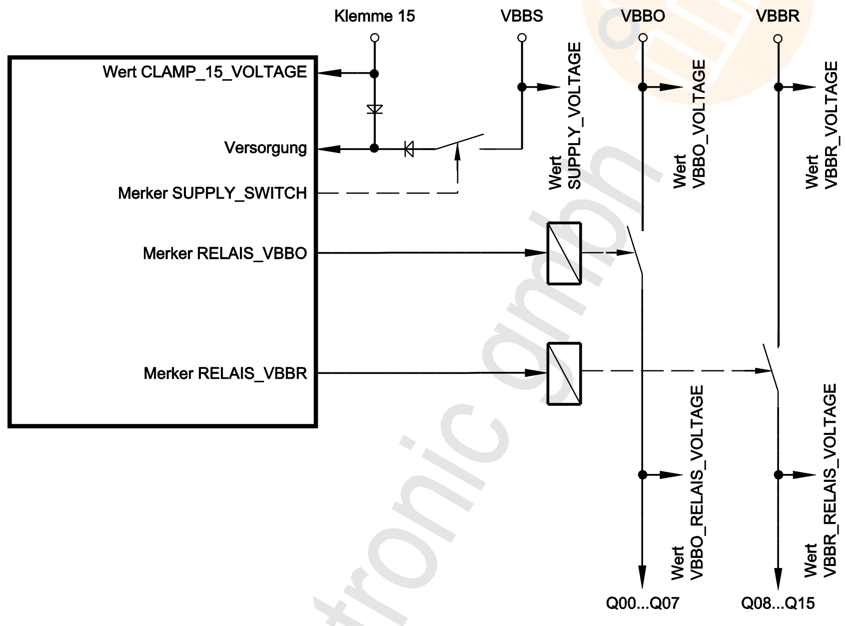
Die Relais werden nur unter folgender Voraussetzung aktiviert:
- Mindestspannung an VBBx = 10 V

UND
- das Bit RELAIS_VBBx = TRUE

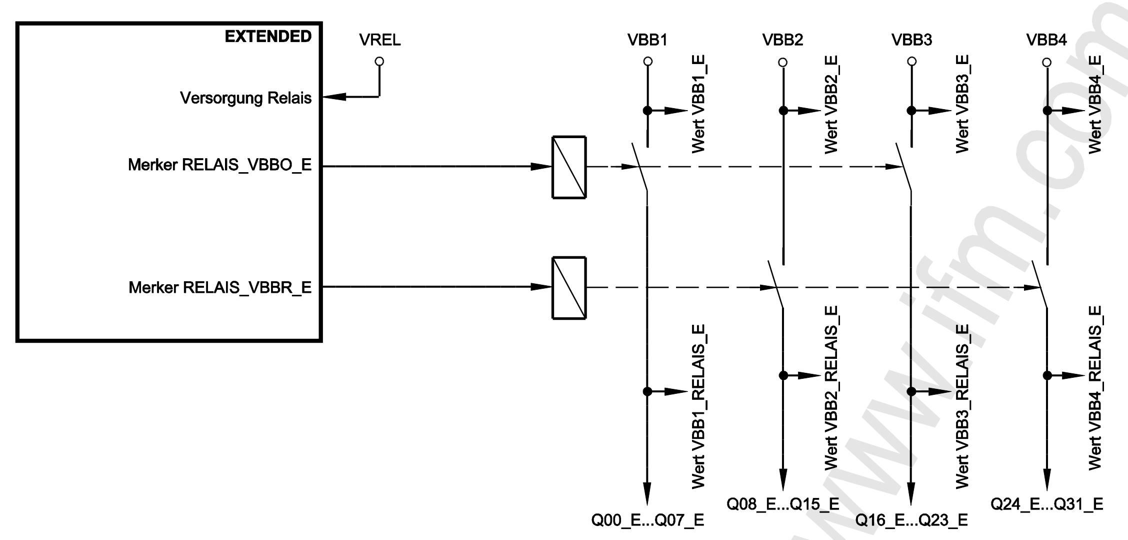
Im aktivierten Zustand legen die Relaiskontakte die Ausgänge an die Klemmenspannung VBBx.

(!) Zugehörige Ausgänge erst ≥ 45 ms nach Einschalten der Relais aktivieren!

Aus den nachfolgenden Prinzipschaltbildern kann die Abhängigkeit der Relais von den anliegenden Signalen und den logischen Zuständen der Systemmerker entnommen werden.



Grafik: Prinzipaufbau der Versorgung und der Relais (Standard-Seite)



Grafik: Prinzipaufbau der Versorgung und der Relais (Extended-Seite)

Verfügbarer Speicher für CR7n32

13131

Physikalischer Speicher	Physikalisch vorhandener FLASH-Speicher (nichtflüchtiger, langsamer Speicher)	6272 kByte
	Physikalisch vorhandener SRAM ¹) (flüchtiger, schneller Speicher)	2216 kByte
	Physikalisch vorhandener EEPROM (nichtflüchtiger, langsamer Speicher)	---
	Physikalisch vorhandener FRAM ²) (nichtflüchtiger, schneller Speicher)	128 kByte
Nutzung des FLASH-Speichers	Speicher reserviert für den Code des Anwendungsprogramms	1280 kByte
	Speicher für Daten außerhalb des Anwendungsprogramms, die vom Anwender beschrieben werden können, wie z.B. Files, Bitmaps, Fonts	128 kByte
	Speicher für Daten außerhalb des Anwendungsprogramms, die vom Anwender mit FBs wie FLASHREAD, FLASHWRITE bearbeitet werden	64 kByte
RAM	Speicher für vom Anwendungsprogramm reservierte Daten im RAM	192 kByte
Remanenter Speicher	Speicher für im Anwendungsprogramm als VAR_RETAIN deklarierte Daten	4 kByte
	Speicher für im Anwendungsprogramm als RETAIN vereinbarte Merker	4 kByte
	Vom Anwender frei verfügbarer remanenter Speicher Der Zugriff erfolgt über die FBs FRAMREAD, FRAMWRITE	16 kByte
	Vom Anwender frei verfügbarer FRAM ²) Der Zugriff erfolgt über Adressoperator	64 kByte

¹) SRAM steht hier allgemein für alle Arten von flüchtigen, schnellen Speichern.
²) FRAM steht hier allgemein für alle Arten von nichtflüchtigen, schnellen Speichern.

4.2.2 Funktionsweise der verzögerten Abschaltung

993

Werden die Controller von der Versorgungsspannung getrennt, werden im Normalfall sofort alle Ausgänge abgeschaltet, keine Eingangssignale mehr eingelesen und die Abarbeitung der Steuerungssoftware (Laufzeitsystem und Anwendungsprogramm) abgebrochen. Dieses geschieht unabhängig davon, in welchem Programmschritt sich der Controller befindet.

Wenn dieses Verhalten nicht gewünscht ist, muss der Controller programmgesteuert abgeschaltet werden. Das ermöglicht nach Abschalten der Zündung zum Beispiel das Sichern von Speicherständen.

Die ClassicController können durch eine entsprechende Beschaltung der Versorgungsspannungs-Eingänge und die Auswertung der zugehörigen Systemmerker, programmgesteuert abgeschaltet werden. Das Prinzipschaltbild im Kapitel ***Hardware-Aufbau*** (→ Seite 128) zeigt schematisch die Zusammenhänge der einzelnen Strompfade.

Klemme VBB15 (32) mit Zündschalter verbinden

2418

Über die Klemme VBB15 (Pin 32) wird die interne Steuerungselektronik initialisiert, wenn an Klemme VBBS (Pin 10) Versorgungsspannung anliegt.

Diese Klemmen VBB15 und VBBS werden intern überwacht. Die anliegende Klemmenspannung VBB15 kann über den Systemmerker CLAMP_15_VOLTAGE überwacht werden. Die anliegende Klemmenspannung VBBS kann über den Systemmerker SUPPLY_VOLTAGE überwacht werden.

Selbsthaltung

2419

Einschalten der Steuerung:

- Der Zündschalter legt Spannung an VBB15 (Klemme 15*).
- Der Systemmerker CLAMP_15_VOLTAGE erkennt die angelegte Spannung und aktiviert den Systemmerker SUPPLY_SWITCH.
- SUPPLY_SWITCH aktiviert die Verbindung zum Potential VBBs.

> Somit ist der Zündschalter überbrückt, die Selbsthaltung der Steuerspannung ist hergestellt.

Ausschalten der Steuerung über Klemme 15:

- Der Systemmerker CLAMP_15_VOLTAGE erkennt das Abschalten der Versorgungsspannung an Klemme VBB15.

► Im Anwendungsprogramm den Systemmerker SUPPLY_SWITCH zurücksetzen.

> Die Selbsthaltung über VBBS (Pin 10) ist aufgehoben und der Controller wird vollständig abgeschaltet.

*) Klemme 15 ist in Fahrzeugen die vom Zündschloss geschaltete Plusleitung.

4.2.3 Relais: wichtige Hinweise!

12978
12976

Zuordnung Relais – Potentiale: → Datenblatt
Max. Summenstrom je Relaiskontakt (= je Ausgangsgruppe): → Datenblatt

ACHTUNG

Gefahr der Zerstörung der Relaiskontakte!
"Klebende" Relaiskontakte können auch im Notfall nicht mehr die Ausgänge von der Versorgung trennen!

Falls VBBs (VBBrel) und Klemme 15 gleichzeitig von der Versorgung getrennt werden, jedoch die Potentiale VBBx an der Versorgung angeschlossen bleiben, dann können die Relais schon abfallen, bevor die Ausgänge vom System deaktiviert werden.

In diesem Fall trennen die Relais **unter Last** die Ausgänge von der Versorgung. Dies schränkt die Lebensdauer der Relais deutlich ein.

► Bei dauerhaftem Anschluss von VBBx an Versorgung:
  • auch VBBs (VBBrel) dauerhaft anschließen und
  • die Ausgänge programmgesteuert mit Hilfe von Klemme 15 abschalten.

Für die SafetyController gilt zusätzlich:

Die Relais können auch in folgendem Fall die Ausgänge unter Last ausschalten:
Ein Ausgang wird als defekt erkannt wegen ...
• defekter Ausgangstreiber oder
• Schluss gegen Versorgung in der Verkabelung nach dem Ausgang

Abhilfe

► Im Anwendungsprogramm dafür sorgen, dass die Steuerung (und somit die betroffenen Relais) nicht öfter als 10-mal vom Maschinenführer wieder in Betrieb genommen werden kann, wenn ein Fehler-Code (→ Kapitel *Fehler-Codes* (→ Seite 392)) mit den folgenden Eigenschaften vom System gemeldet wurde:
  • Fehlerursache = 0x0A (Safety-Diagnose am Ausgang)
  • Fehlerquelle = 0x40...0x4F (Ausgänge Standard-Seite)
  • Fehlerklasse = 0x02 (schwerer Fehler)

Ansonsten kann ifm electronic nicht garantieren, dass die Relais im Fehlerfall die Ausgänge von der Versorgung trennen können.

14179

HINWEIS

Damit das System das Relais im geöffneten Zustand überprüft, muss mindestens einmal innerhalb von 24 Stunden Folgendes geschehen:
• der SafetyController startet neu oder
• die Relais werden für länger als 100 ms geöffnet.

4.2.4 Überwachungskonzept

Inhalt

991

Die Steuerung überwacht die Versorgungsspannungen und die System-Fehlermerker. Je nach Zustand schaltet die Steuerung die internen Relais oder schaltet den Controller ab.

Überwachung der Versorgungsspannungen

6752

Wir unterscheiden 2 Szenarien:

Klemmenspannung VBBx fällt unter den Grenzwert von 5,25 V

3932

> Die Steuerung erkennt Unterspannung. Die von der Klemmenspannung VBBx versorgten Ausgänge werden deaktiviert.

> Erholt sich die Klemmenspannung und befindet sich wieder im regulären Bereich, werden die nicht-sicheren Ausgänge wieder aktiviert.

⚠ WARNUNG

Gefährlicher Wiederanlauf möglich!
Gefahr von Personenschaden! Gefahr von Sachschaden an der Maschine/Anlage!

Wird ein Ausgang im Fehlerfall hardwaremäßig abgeschaltet, ändert sich der durch das Anwendungsprogramm erzeugte logische Zustand dadurch nicht.

► Abhilfe:
- Die Ausgänge zunächst im Anwendungsprogramm logisch zurücksetzen!
- Fehler beseitigen!
- Ausgänge situationsabhängig wieder setzen.

► Die sicheren Ausgänge durch Zurücksetzen des Fehler-Codes in der Anwendung wieder aktivieren.
Der Fehler-Code kann nur zurückgesetzt werden, wenn die Versorgungsspannung an der Klemme wieder im regulären Bereich ist.

Versorgungsspannung VBBS fällt unter den Grenzwert von 8 V

3956

> Die Steuerung läuft weiter, bis die Spannung so weit gefallen ist, dass die daraus erzeugten internen Spannungen einbrechen.

! Unterhalb von 8 V werden keine Retain-Daten gespeichert. → Merker RETAIN_WARNING

> Brechen die internen Spannungen ein, geht der Controller in den Reset.
Die Ausführung von Laufzeitsystem und Anwendungsprogramm wird abgebrochen.
Dies geschieht unabhängig davon, in welchem Programmschritt sich die Steuerung befindet.

> Ein Wiederanlauf der Steuerung erfolgt erst, wenn die Versorgungsspannungen wieder oberhalb des Grenzwerts sind.

Überwachungs- und Sicherungsmechanismen

14085

→ *Überwachungs- und Sicherungsmechanismen* (→ Seite 45)
(im Kapitel *Hinweise für sicherheitsrelevante Anwendungen* (→ Seite 15))

Referenzspannungsausgang

13934

Der Referenzspannungsausgang dient der Versorgung von Sensoren mit einer stabilen Spannung, die nicht den Schwankungen der Versorgungsspannung unterworfen ist.

13402

ACHTUNG

Referenzspannungsausgang kann beschädigt werden!

► Von außen KEINE Spannung anlegen!

Über die binären Systemvariablen REFERENCE_VOLTAGE_5 oder REFERENCE_VOLTAGE_10 wird die Spannung am Referenzspannungsausgang [V_{REF} OUT] eingestellt:

REFERENZ_VOLTAGE_10	REFERENZ_VOLTAGE_5	Referenzspannung [V_{REF} OUT]
FALSE	FALSE	0 V
FALSE	TRUE	5 V
TRUE	FALSE	10 V
TRUE	TRUE	0 V

► Wenn Referenzspannungsausgang = 10 V gewählt:
die Steuerung mit mindestens 13 V versorgen!

► Überwachen der Spannung am Referenzspannungsausgang mit Systemvariable REF_VOLTAGE.

4.2.5 Eingänge (Technologie)

Inhalt

14090

Analog-Eingänge

2426

Die Analog-Eingänge können über das Anwendungsprogramm konfiguriert werden. Der Messbereich kann zwischen folgenden Bereichen umgeschaltet werden:

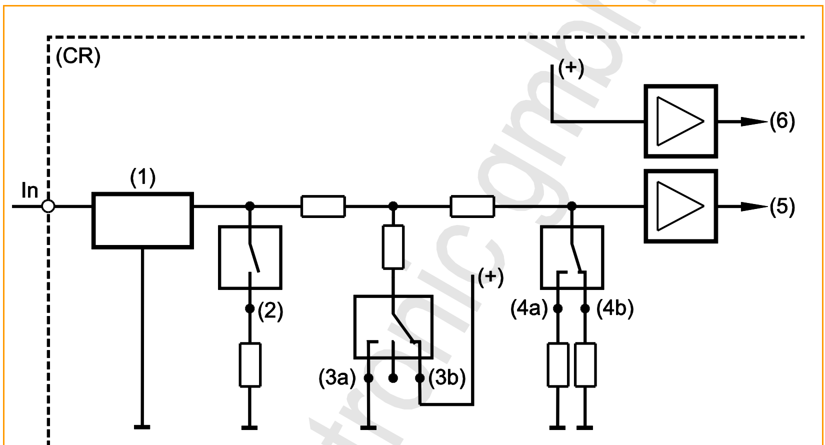
- Stromeingang 0...20 mA
- Spannungseingang 0...10 V
- Spannungseingang 0...32 V

Die Spannungsmessung kann auch ratiometrisch erfolgen (0...1000 ‰, über FBs einstellbar). Das bedeutet, ohne zusätzliche Referenzspannung können Potentiometer oder Joysticks ausgewertet werden. Ein Schwanken der Versorgungsspannung hat auf diesen Messwert keinen Einfluss.

Alternativ kann ein Analog-Kanal auch binär ausgewertet werden.

> ! Bei ratiometrischer Messung müssen die angeschlossenen Sensoren mit VBBs des Geräts versorgt werden. Dadurch werden Fehlmessungen durch Spannungsverschiebungen vermieden.

8971



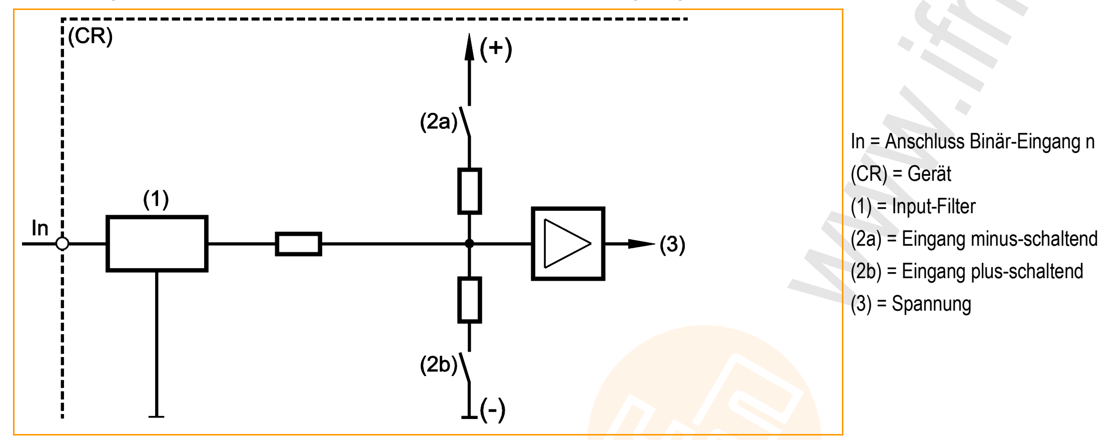
In = Anschluss Multifunktions-Eingang n
(CR) = Gerät
(1) = Eingangsfilter
(2) = analoge Strommessung
(3a) = Binär-Eingang plus-schaltend
(3b) = Binär-Eingang minus-schaltend
(4a) = analoge Spannungsmessung 0...10 V
(4b) = analoge Spannungsmessung 0...32 V
(5) = Spannung
(6) = Referenz-Spannung

Grafik: Prinzipschaltung Multifunktions-Eingang

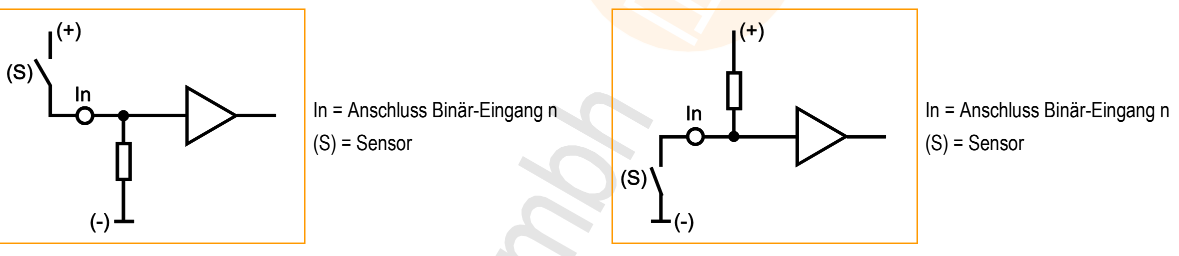
Binär-Eingänge

1015
7345

Je nach Gerät können auch die Binär-Eingänge unterschiedlich konfiguriert werden. Neben den Schutzmechanismen gegen Störungen werden die Binär-Eingänge intern über eine Analogstufe ausgewertet. Das ermöglicht die Diagnose der Eingangssignale. Im Anwendungsprogramm steht das Schaltsignal aber direkt als Bit-Information zur Verfügung.



Grafik: Prinzipschaltung Binär-Eingang minus-schaltend / plus-schaltend für negative und positive Gebersignale



Prinzipschaltung Binär-Eingang plus-schaltend (BL)
für positives Sensorsignal:
Eingang = offen ⇨ Signal = Low (Supply)

Prinzipschaltung Binär-Eingang minus-schaltend (BH)
für negatives Sensorsignal:
Eingang = offen ⇨ Signal = High (GND)

Bei einem Teil dieser Eingänge (→ Datenblatt) kann das Potential gewählt werden, gegen das geschaltet wird.

Eingangsgruppe I0 (I00...I15)

12280

Bei diesen Eingängen handelt es sich um eine Gruppe von Multifunktionskanälen.

Jeder einzelne dieser Eingänge ist wahlweise wie folgt konfigurierbar:
- analoger Eingang 0...20 mA
- analoger Eingang 0...10 V
- analoger Eingang 0...32 V
- binärer Eingang, minus-schaltend (BH) für negatives Gebersignal
- binärer Eingang plus-schaltend (BL) für positives Gebersignal
- schneller Eingang für z.B. Inkrementalgeber und Frequenz- oder Periodendauermessung

→ Kapitel ***Mögliche Betriebsarten Ein-/Ausgänge*** (→ Seite 436)

Diagnosefähige Sensoren nach NAMUR können ausgewertet werden.

Alle Eingänge zeigen das gleiche Verhalten bei Funktion und Diagnose.

Detaillierte Beschreibung → Kapitel ***Adressbelegung Ein-/Ausgänge*** (→ Seite 430)

Sichere Eingänge

12249

→ Kapitel ***Eingänge für Sicherheitsfunktionen*** (→ Seite 55)
(im Kapitel ***Hinweise für sicherheitsrelevante Anwendungen*** (→ Seite 15))

Eingangsgruppe I1 (I00_E...I15_E)

13531

Prinzipiell gelten die gleichen Aussagen wie für die Eingangsgruppe I0.
Abweichungen:
- Die Eingänge sind für sichere Anwendungen nicht geeignet.
- Die symbolischen Adressen der Eingänge lauten Inn_*E*.
- Die symbolischen Adressen der Filter lauten Inn_FILTER_*E*
- Die symbolischen Adressen der digitalen Filter lauten Inn_DFILTER_*E*
- Die symbolischen Adressen der anderen Merker enden ebenfalls auf '_*E*'.

4.2.6 Ausgänge (Technologie)

Inhalt

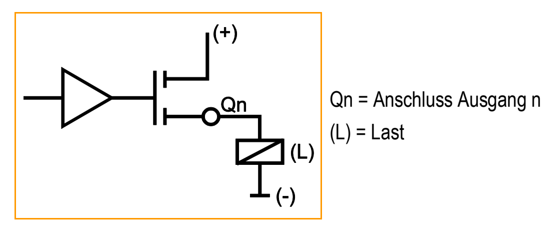
14093

Binär-Ausgänge

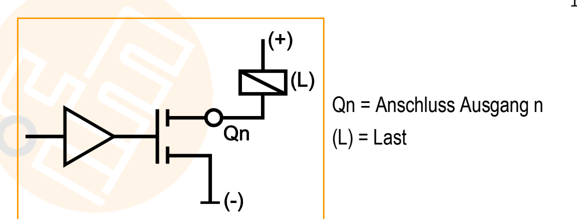
14094

Bei den Geräte-Ausgängen sind folgende Betriebsarten möglich (→ Datenblatt):
- binärer Ausgang, plus-schaltend (BH) mit/ohne Diagnosefunktion
- binärer Ausgang plus-schaltend (BL) ohne Diagnosefunktion

15450



Prinzipschaltung Binär-Ausgang plus-schaltend (BH) für positives Ausgangssignal



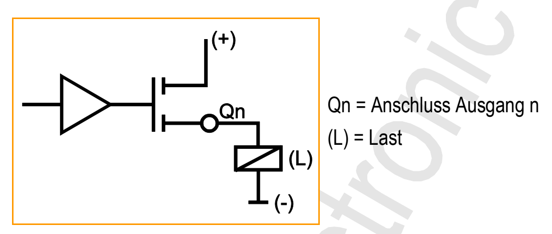
Prinzipschaltung Binär-Ausgang minus-schaltend (BL) für negatives Ausgangssignal

PWM-Ausgänge

14095

Bei den Geräte-Ausgängen sind folgende Betriebsarten möglich (→ Datenblatt):
- PWM-Ausgang, plus-schaltend (BH) ohne Diagnosefunktion

15451



Prinzipschaltung Binär-Ausgang plus-schaltend (BH) für positives Ausgangssignal

Schutzfunktionen der Ausgänge

15248

Die Ausgänge dieses Geräts sind in Grenzen gegen Überlast und Kurzschluss geschützt.

Ein Ausfall der Software oder des gesamten Systems durch Kurzschluss oder Überlast sollte ausgeschlossen sein.

Definition: Überlast

15249

Überlast kann nur an einem Ausgang mit Strommessung erkannt werden.

Überlast ist definiert als ...
"nominaler Maximalstrom laut Datenblatt + 12,5 %".

Definition: Kurzschluss

15250

Ein Kurzschluss kann an allen diagnosefähigen Ausgängen erkannt werden und ist wie folgt definiert:

Kurzschluss ist definiert als ...
"Absinken der Ausgangsspannung unter 88 % (± 2,5 % vom gemessenen Wert) der zugehörigen Versorgungsspannung."

> Ein Schluss gegen Masse kann nur erkannt werden bei Ausgang = TRUE.

Reaktion der Ausgänge auf Überlast oder Kurzschluss

15251

Eigenschutz des Ausgangs

15333

Unabhängig von der Betriebsart des Ausgangs und der Fehlererkennung schützt sich die Hardware selbst. Bei zu hoher thermischer Belastung (durch Kurzschluss oder Überlast) beginnt der Ausgangstreiber zu takten.

[!] Bei zu lange andauerndem Takten des Ausgangs (mehrere Stunden) kann der Treiber beschädigt werden!

Wir empfehlen deshalb:
Diagnosefähige Ausgänge des Geräts unbedingt mit folgenden Einstellungen betreiben, da hier die Software zusätzlich die Treiber durch Abschalten schützt:
- FB ***SET_OUTPUT_MODE*** (→ Seite 313) > Eingang DIAGNOSTICS = TRUE und
- FB SET_OUTPUT_MODE > Eingang PROTECTION = TRUE

Reaktion abhängig von Betriebsart des Ausgangs

15479

Im Falle von Überlast oder Kurzschluss hängt das Verhalten des Ausgangs von dessen Betriebsart ab (→ FB ***SET_OUTPUT_MODE*** (→ Seite 313) > Eingänge DIAGNOSTICS und PROTECTION):

- DIAGNOSTICS = FALSE und PROTECTION = FALSE:
  > der Ausgang wird weiter betrieben.
- DIAGNOSTICS = TRUE und PROTECTION = FALSE:
  > Fehler wird erkannt und vom FB SET_OUTPUT_MODE am Ausgang ERROR gemeldet.
  Das hängt vom Ausgangstyp und dem Strom oder der Spannung am Ausgang ab.
  Der Programmierer kann im Programm auf den Fehler reagieren.
- DIAGNOSTICS = TRUE und PROTECTION = TRUE:
  > Fehler wird erkannt und vom FB SET_OUTPUT_MODE am Ausgang ERROR gemeldet.
  > Der betreffende Ausgang wird abgeschaltet.
  > [!] Der logische Zustand des Ausgangs bleibt davon unverändert!

Reaktion bei Einsatz von PWM oder OUTPUT_CURRENT_CONTROL

15480

Anders verhält es sich bei Einsatz der FBs PWM oder OUTPUT_CURRENT_CONTROL:
Hier gibt es keine Diagnose. Der →Eigenschutz des Eingangs wird aktiv.

► Bei Ausgängen mit Stromrücklesung:
Im Anwendungsprogramm den typischen Strom für den Ausgang abfragen!
Hier ist der Anwendungsprogrammierer verantwortlich, auf das Ereignis zu reagieren.

Reaktion der Ausgänge für Sicherheitsfunktionen

15348

→ Kapitel ***Ausgänge für Sicherheitsfunktionen*** (→ Seite 65)

Ausgangsgruppe Q0 (Q00...Q15)

12295

Bei diesen Ausgängen handelt es sich um eine Gruppe von Multifunktionskanälen.

Jeder einzelne dieser Ausgänge ist wahlweise wie folgt konfigurierbar:
- binärer Ausgang, plus-schaltend (BH), teilweise auch minus-schaltend (BL)
- analoger Ausgang, stromgeregelt (PWMi)
- analoger Ausgang mit Pulsweitenmodulation (PWM), teilweise als H-Brücke

→ Kapitel *Mögliche Betriebsarten Ein-/Ausgänge* (→ Seite 436)

► Die Konfiguration jedes einzelnen Ausgangs erfolgt über das Anwendungsprogramm:
→ FB *SET_OUTPUT_MODE* (→ Seite 313) > Eingang MODE
Lastströme anzeigen → FB *OUTPUT_CURRENT* (→ Seite 330)
PWM-Ausgang: → FB *PWM1000* (→ Seite 333)
H-Brücke steuern → FB *OUTPUT_BRIDGE* (→ Seite 326)

► Strommessbereich konfigurieren für die Ausgänge Q00…Q03 und Q08…Q11 (wahlweise 2 A oder 4 A):
→ FB SET_OUTPUT_MODE > Eingang CURRENT_RANGE

Bei Einsatz der H-Brücke wird die Stromregelung nicht unterstützt.

HINWEIS

Um die internen Messwiderstände zu schützen, sollte der Überlastschutz immer aktiv sein (voreingestellt). Je nach gewähltem Strommessbereich besteht Schutz ab 2,25 A oder ab 4,5 A.
Die Funktion wird **nicht** im PWM-Modus unterstützt.
Die Funktion kann bei Bedarf abgeschaltet werden.

Zu den Grenzwerten unbedingt das Datenblatt beachten!

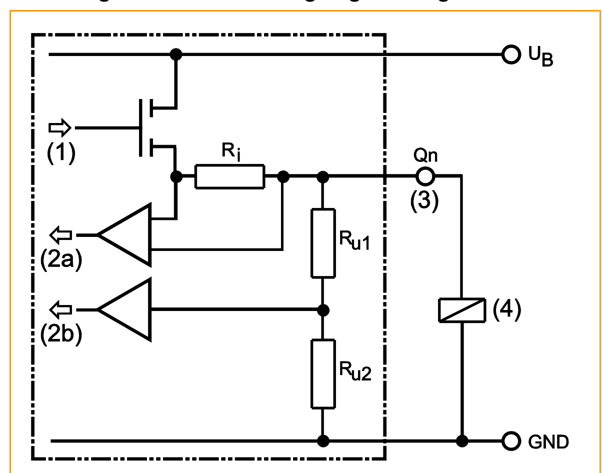
Die Leiterbruch- und die Kurzschlusserkennung sind aktiv, wenn der Ausgang **ein**geschaltet ist.

► Bei Verwendung von gegen Masse schaltenden Ausgängen darf die Versorgungsspannung an der angeschlossenen Last nicht höher sein als die Versorgungsspannung(en) der Ausgangsgruppe(n)!

Diagnose: binäre Ausgänge (via Strom- und Spannungsmessung)

19461
19462

Die Diagnose dieser Ausgänge erfolgt über eine interne Strom- und Spannungsmessung im Ausgang:



Grafik: Prinzipschaltung
(1) Ausgangskanal
(2a) Rücklesekanal für Diagnose via Strommessung
(2b) Rücklesekanal für Diagnose via Spannungsmessung
(3) Anschluss Ausgang
(4) Last

Diagnose: Überlast

19437
15249

Überlast kann nur an einem Ausgang mit Strommessung erkannt werden.

Überlast ist definiert als ...
"nominaler Maximalstrom laut Datenblatt + 12,5 %".

Diagnose: Leiterbruch

19400

Eine Leiterbruch-Erkennung erfolgt über den Rücklesekanal. Bei geschaltetem Ausgang (Qn=TRUE) wird dann ein Leiterbruch erkannt, wenn über den Widerstand R_i kein Strom fließt (keine Spannung abfällt). Ohne den Leiterbruch fließt durch den Längswiderstand R_i der Laststrom und erzeugt damit einen Spannungsabfall, der über den Rücklesekanal ausgewertet wird.

Diagnose: Kurzschluss

19405

Eine Kurzschluss-Erkennung erfolgt über den Rücklesekanal. Bei geschaltetem Ausgang (Qn=TRUE) wird dann ein Kurzschluss gegen GND erkannt, wenn der Rücklesekanal auf LOW-Potential (GND) gezogen wird.

Diagnose: Querschluss

19475

Diese Diagnose erfolgt nur für sicherheitsrelevante Ausgänge:
→ Kapitel ***Querschluss erkennen*** (→ Seite 68)
(im Kapitel ***Hinweise für sicherheitsrelevante Anwendungen*** (→ Seite 15))

Diagnose: Ausgangstreiber-Baustein defekt

19460

Diese Diagnose erfolgt nur für sicherheitsrelevante Ausgänge:

Einen defekten Ausgangstreiber-Baustein ermittelt das Gerät anhand von Strommessung und Spannungsmessung.

► Im FB ***SET_OUTPUT_MODE*** (→ Seite 313): für jeden sicherheitsrelevanten Ausgang...
  • den Parameter SAFETY = TRUE setzen!
  • den Parameter DIAGNOSTICS = TRUE setzen!
  (!) Falls SAFETY = TRUE und DIAGNOSTICS = FALSE ➯ Fehlermeldung!

Sichere Ausgänge

1992

→ Kapitel ***Ausgänge für Sicherheitsfunktionen*** (→ Seite 65)
(im Kapitel ***Hinweise für sicherheitsrelevante Anwendungen*** (→ Seite 15))

Ausgangsgruppe Q1 (Q00_E...Q15_E)

13527

Prinzipiell gelten die gleichen Aussagen wie für die Ausgangsgruppe Q0.
Abweichungen:

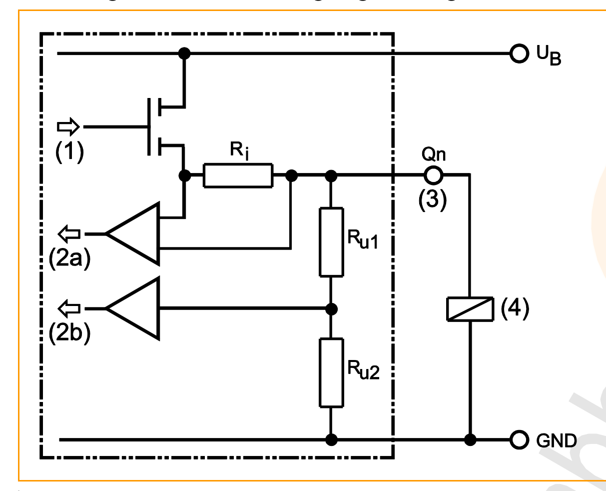
- Die Ausgänge sind für sichere Anwendungen nicht geeignet.
- Die symbolischen Adressen der Ausgänge lauten Qnn_*E*.
- Die symbolischen Adressen der anderen Merker enden ebenfalls auf '_*E*'.

Zu den Grenzwerten unbedingt das Datenblatt beachten!

Diagnose: binäre Ausgänge (via Strom- und Spannungsmessung)

19461
19462

Die Diagnose dieser Ausgänge erfolgt über eine interne Strom- und Spannungsmessung im Ausgang:



Grafik: Prinzipschaltung
(1) Ausgangskanal
(2a) Rücklesekanal für Diagnose via Strommessung
(2b) Rücklesekanal für Diagnose via Spannungsmessung
(3) Anschluss Ausgang
(4) Last

Diagnose: Überlast

19437
15249

Überlast kann nur an einem Ausgang mit Strommessung erkannt werden.

Überlast ist definiert als ...
"nominaler Maximalstrom laut Datenblatt + 12,5 %".

Diagnose: Leiterbruch

19400

Eine Leiterbruch-Erkennung erfolgt über den Rücklesekanal. Bei geschaltetem Ausgang (Qn=TRUE) wird dann ein Leiterbruch erkannt, wenn über den Widerstand R_i kein Strom fließt (keine Spannung abfällt). Ohne den Leiterbruch fließt durch den Längswiderstand R_i der Laststrom und erzeugt damit einen Spannungsabfall, der über den Rücklesekanal ausgewertet wird.

Diagnose: Kurzschluss

19405

Eine Kurzschluss-Erkennung erfolgt über den Rücklesekanal. Bei geschaltetem Ausgang (Qn=TRUE) wird dann ein Kurzschluss gegen GND erkannt, wenn der Rücklesekanal auf LOW-Potential (GND) gezogen wird.

Ausgangsgruppe Q2 (Q16_E...Q31_E)

13529

Prinzipiell gelten die gleichen Aussagen wie für die Ausgangsgruppe Q0.
Abweichungen:

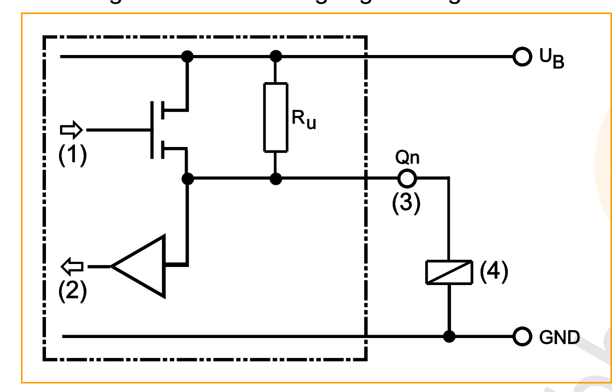
- Die Ausgänge sind für sichere Anwendungen nicht geeignet.
- Die symbolischen Adressen der Ausgänge lauten Qnn_*E*.
- Die symbolischen Adressen der anderen Merker enden ebenfalls auf '_*E*'.
- Die Ausgänge sind maximal mit 2 A belastbar (fest eingestellt).
- Die Ausgänge sind auf binär plus-schaltend fest eingestellt.

Zu den Grenzwerten unbedingt das Datenblatt beachten!

Diagnose: binäre Ausgänge (via Spannungsmessung)

19403
19397

Die Diagnose dieser Ausgänge erfolgt über eine interne Spannungsmessung im Ausgang:



Grafik: Prinzipschaltung
(1) Ausgangskanal
(2) Rücklesekanal für Diagnose
(3) Anschluss Ausgang n
(4) Last

Diagnose: Überlast

19448

Die Ausgänge haben keine Strommessung, keine Überlasterkennung.

Diagnose: Leiterbruch

19404

Eine Leiterbruch-Erkennung erfolgt über den Rücklesekanal. Bei gesperrtem Ausgang (Qn=FALSE) wird dann ein Leiterbruch erkannt, wenn der Widerstand R_u den Rücklesekanal auf HIGH-Potential (VBB) zieht. Ohne den Leiterbruch würde die niederohmige Last (R_L < 10 kOhm) LOW (logisch 0) erzwingen.

Diagnose: Kurzschluss

19405

Eine Kurzschluss-Erkennung erfolgt über den Rücklesekanal. Bei geschaltetem Ausgang (Qn=TRUE) wird dann ein Kurzschluss gegen GND erkannt, wenn der Rücklesekanal auf LOW-Potential (GND) gezogen wird.

4.2.7 Hinweise zur Anschlussbelegung

1426

Die Anschlussbelegungen (→ Montageanleitungen der Geräte, Kapitel "Anschlussbelegung") beschreiben die Standard-Gerätekonfigurationen. Die Anschlussbelegung dient der Zuordnung der Ein- und Ausgangskanäle zu den IEC-Adressen und den Geräteanschlussklemmen.

Die einzelnen Kürzel haben folgende Bedeutung:

A	Analog-Eingang
BH	Binärer highside-Eingang: minus-schaltend für negatives Sensorsignal Binärer highside-Ausgang: plus-schaltend für positives Ausgangssignal
BL	Binärer lowside-Eingang: plus-schaltend für positives Sensorsignal Binärer lowside-Ausgang: minus-schaltend für negatives Ausgangssignal
CYL	Eingang Periodendauermessung
ENC	Eingang Drehgebersignale
FRQ	Frequenzeingang
H-Bridge	Ausgang mit H-Brücken-Funktion
PWM	**P**uls**w**eiten-**m**oduliertes Signal
PWMi	PWM-Ausgang mit Strommessung
IH	Impuls-/Zählereingang, highside, minus-schaltend für negatives Sensorsignal
IL	Impuls-/Zählereingang, lowside, plus-schaltend für positives Sensorsignal
R	Rücklesekanal für einen Ausgang

Zuordnung der Ein-/Ausgangskanäle: → Katalog, Montageanleitung oder Datenblatt

4.2.8 Sicherheitshinweise zu Reed-Relais

7348

Beim Einsatz von nichtelektronischen Schaltern Folgendes beachten:

> ❗ Kontakte von Reed-Relais können (reversibel) verkleben, wenn sie ohne Vorwiderstand an den Geräte-Eingängen angeschlossen werden.

- ► **Abhilfe:** Vorwiderstand zum Reed-Relais installieren:
  Vorwiderstand = max. Eingangsspannung / zulässiger Strom im Reed-Relais
  Beispiel: 32 V / 500 mA = 64 Ohm
- ► Der Vorwiderstand darf 5 % des Eingangswiderstands RE des Geräte-Eingangs (→ Datenblatt) nicht überschreiten. Sonst wird das Signal nicht als TRUE erkannt.
  Beispiel:
  RE = 3 000 Ohm
  ⇨ max. Vorwiderstand = 150 Ohm



4.2.9 Rückspeisung bei extern beschalteten Ausgängen

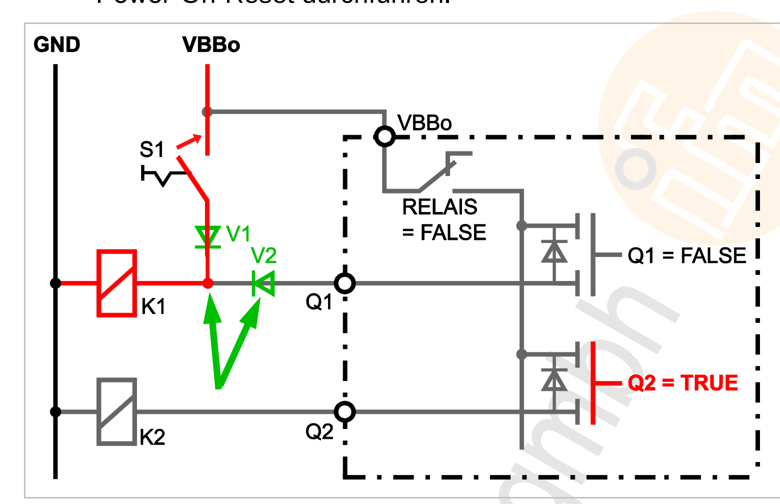
13932

► Von außen keine Spannung an die Ausgänge legen!

> Sobald RELAIS = FALSE:
Die interne Geräteüberwachung prüft die Spannungen auf den Stromschienen hinter den Relais.
Wird dabei eine Spannung gemessen von > ½ VBBx:
 • die Steuerung meldet einen fatalen Fehler,
 • die Steuerung geht in den sicheren Zustand.

ⓘ Sicherer Zustand = das Laufzeitsystem stoppt die Steuerung (fataler Fehler):
 • alle Ausgänge werden abgeschaltet
 • die Abarbeitung der Software wird angehalten
 • es ist keine Kommunikation mehr möglich
 • die LED erlischt.

► Für Neustart des Geräts:
 • Fehlerursache beseitigen
 • Power-On-Reset durchführen.



Grafik: Beispiel Beschaltung mit Sperrdioden wegen Gefahr der Rückspeisung

Abhilfe:
Sperrdioden V1 und V2 einsetzen (→ grüne Pfeile)!

Erfolg:
Wenn RELAIS = FALSE, dann geht die Steuerung nicht in den fatalen Fehler, wenn der Kontakt S1 geschlossen wird.

ⓘ HINWEIS

Abhilfe bei extern beschalteten Ausgängen

► Die extern beschalteten Ausgänge so über Dioden entkoppeln, dass keine externe Spannung an die Ausgangsklemme der Steuerung geschaltet werden kann!

4.2.10 Status-LED

13505

Die Betriebszustände werden durch die integrierte Status-LED (Voreinstellung) angezeigt.

LED-Farbe	Blinkfrequenz	Beschreibung
aus	konstant aus	keine Betriebsspannung oder Fatal Error
Orange	konstant ein	kein Laufzeitsystem geladen (ohne TEST-Pin)
Grün / schwarz	5 Hz	kein Laufzeitsystem geladen (mit TEST-Pin)
Grün / schwarz	2 Hz	Anwendung RUN
Grün	konstant ein	Anwendung STOP
Rot / schwarz	2 Hz	Anwendung RUN mit Fehler
rot	konstant ein	Fehler im Zustand Anwendung STOP Unterspannung < 9 V nach Initialisierung
Orange	kurzzeitig ein	Initialisierung

Die Betriebszustände STOP und RUN können vom Programmiersystem geändert werden.

LED im Anwendungsprogramm steuern

13142

Bei diesem Gerät kann die Status-LED auch durch das Anwendungsprogramm gesetzt werden. Dazu dienen folgende Systemvariablen (→ Kapitel ***Systemmerker*** (→ Seite 420)):

Systemmerker (Symbolname)	Typ	Beschreibung
LED	WORD	LED-Farbe für "LED eingeschaltet": 0x0000 = LED_GREEN (voreingestellt) 0x0001 = LED_BLUE 0x0002 = LED_RED 0x0003 = LED_WHITE 0x0004 = LED_BLACK 0x0005 = LED_MAGENTA 0x0006 = LED_CYAN 0x0007 = LED_YELLOW
LED_X	WORD	LED-Farbe für "LED ausgeschaltet": 0x0000 = LED_GREEN 0x0001 = LED_BLUE 0x0002 = LED_RED 0x0003 = LED_WHITE 0x0004 = LED_BLACK (voreingestellt) 0x0005 = LED_MAGENTA 0x0006 = LED_CYAN 0x0007 = LED_YELLOW
LED_MODE	WORD	LED-Blinkfrequenz: 0x0000 = LED_2HZ (blinkt mit 2 Hz; voreingestellt) 0x0001 = LED_1HZ (blinkt mit 1 Hz) 0x0002 = LED_05HZ (blinkt mit 0,5 Hz) 0x0003 = LED_0HZ (leuchtet dauernd mit Wert in LED)

HINWEIS

► Im Anwendungsprogramm NICHT die LED-Farbe ROT verwenden.

> Im Fehlerfall wird die LED-Farbe ROT durch das Laufzeitsystem gesetzt.
ABER: Werden die Farben und/oder Blinkmodi im Anwendungsprogramm geändert, gilt die obige Tabelle der Voreinstellung nicht mehr.

4.3 Schnittstellen-Beschreibung

Inhalt

14098

4.3.1 Serielle Schnittstelle

14099

Dieses Gerät bietet eine serielle Schnittstelle.

Grundsätzlich kann die serielle Schnittstelle mit folgenden Funktionen genutzt werden:

- Programm-Download
- Debugging
- freie Nutzung in der Anwendung

HINWEIS

Voreingestellt steht die serielle Schnittstelle dem Anwender nicht zur Verfügung, da sie für den Programm-Download und das Debugging genutzt wird.

Setzt der Anwender das Systemmerkerbit SERIAL_MODE=TRUE, dann kann die Schnittstelle frei genutzt werden. Ein Debugging des Anwendungsprogramms ist dann nur noch über eine der 4 CAN-Schnittstellen oder über USB möglich.

Anschlüsse und Daten → Datenblatt

4.3.2 USB-Schnittstelle

14100

Dieses Gerät bietet eine USB-Schnittstelle für den Programm-Download und das Debugging.

Anschlüsse und Daten → Datenblatt
USB-Treiber auf dem PC installieren → Montageanleitung / Betriebsanleitung
Einstellungen in CODESYS für [Online] > [Kommunikationsparameter...] via USB:

Gerät	Laufzeitsystem-Version	Parameter	Wert
CR7n32	≤ V 01.00.04	Baudrate	115200
CR7n32	≥ V 01.00.05	Baudrate	4800...57600
CR0032	≤ V 02.01.05	Baudrate	115200
CR0032	≥ V03.00.01	Baudrate	4800...57600
CR0033, CR0133	≤ V 02.00.01	Baudrate	115200
CR0033, CR0133	≥ V 02.00.02	Baudrate	4800...57600
CR0232, CR0233	alle	Baudrate	115200
CR0234, CR0235	alle	Baudrate	4800...57600
CR0n3n, CR7n32	alle	Motorola byteorder	No
CR0n3n, CR7n32	alle	Flow Control	On

4.3.3 CAN-Schnittstellen

Inhalt

14101

Anschlüsse und Daten → Datenblatt

CAN: Schnittstellen und Protokolle

13809
14588

Die Controller werden je nach Aufbau der Hardware mit mehreren CAN-Schnittstellen ausgerüstet. Grundsätzlich können alle Schnittstellen unabhängig voneinander mit folgenden Funktionen genutzt werden:

- Layer 2: CAN auf Ebene 2 (→ Kapitel ***Bausteine: CAN Layer 2*** (→ Seite 202))
- CANopen-Master (→ Kapitel ***Bausteine: CANopen-Master*** (→ Seite 220))
- CANopen-Slave (→ Kapitel ***Bausteine: CANopen-Slave*** (→ Seite 230))
- CANsafety*) (→ Kapitel ***Bausteine: Daten sicher übertragen*** (→ Seite 213))
- CAN-Netzwerkvariablen (via CODESYS)
- SAE J1939*) (für Antriebsmanagement, → Kapitel ***Bausteine: SAE J1939*** (→ Seite 243))
- Buslast-Erkennung
- Errorframe-Zähler
- Download-Schnittstelle
- 100 % Buslast ohne Paketverlust

*)

HINWEIS

CANsafety benötigt 2 mit 11 Bit betriebene CAN-Schnittstellen gleichzeitig.

► Bei Verwendung von CANsafety dasselbe CAN-Schnittstellenpaar nicht gleichzeitig für SAE J1939 oder Extended Frames (29 Bit) einsetzen!

13162

In diesem **ecomat*mobile***-Gerät sind folgende CAN-Schnittstellen und CAN-Protokolle verfügbar:

CAN-Schnittstelle	CAN 1	CAN 2	CAN 3	CAN 4
voreingestellte Download-ID	ID 127	ID 126	ID 125	ID 124
CAN-Protokolle	CAN Layer 2	CAN Layer 2	CAN Layer 2	CAN Layer 2
	CANopen	CANopen	CANopen	CANopen
	CANsafety		CANsafety	
	SAE J1939	SAE J1939	SAE J1939	SAE J1939

Standard-Baudrate = 125 kBit/s

Welche CANopen-fähige Schnittstelle mit welchem CANopen-Protokoll arbeitet, entscheidet die Reihenfolge, mit der Sie in der Steuerungskonfiguration die Unterelemente anhängen:
CODESYS > [Steuerungskonfiguration] > [CR7132 Configuration Vxx] > [Unterelement anhängen] > [CANopen Master] oder [CANopen Slave]

CAN: Hardware

14103

Topologie

1244

Das CAN-Netzwerk wird als Linienstruktur aufgebaut. Stichleitungen sind in eingeschränktem Umfang zulässig.

Weitere Möglichkeit:
• sternförmiger Bus (z.B. Zentralverrieglung).

HINWEIS

Verfälschen der Signalqualität wegen Signal-Echos an den Leitungsenden verhindern:

► Die CAN-Buslinie an ihren beiden Enden jeweils mit einem Abschlusswiderstand von jeweils ≥ 120 Ω abschließen!

Die Geräte der **ifm electronic gmbh**, die mit einem CAN-Interface ausgestattet sind, haben grundsätzlich keine Abschlusswiderstände.

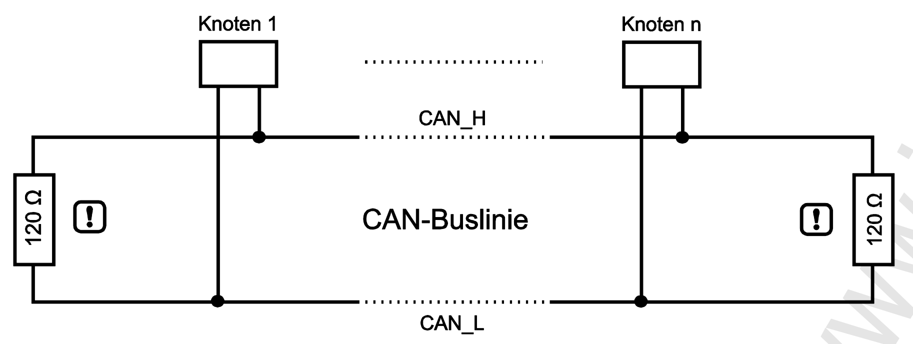
Zusammen mit den Abschlusswiderständen soll der Gesamtwiderstand (gemessen zwischen CAN_H und CAN_L) der spannungslosen CAN-Buslinie etwa 60±5 Ω betragen.

Stichleitungen und sternförmiger Bus haben den Nachteil, dass der Wellenwiderstand schwer zu bestimmen ist. Im schlimmsten Fall funktioniert der Bus nicht mehr.

Netzaufbau

1178

Die Norm ISO 11898 setzt einen Aufbau des CAN-Netzes mit einer Linienstruktur voraus.



Grafik: CAN-Netzaufbau Linienstruktur

Der Innenwiderstand einer CAN-Schnittstelle beträgt etwa 40...45 kΩ. Bei 32 Geräten am CAN-Bus beträgt der resultierende Widerstand im Netzwerk nur noch 1,25...1,4 kΩ.

HINWEIS

Verfälschen der Signalqualität wegen Signal-Echos an den Leitungsenden verhindern:

► Die CAN-Buslinie an ihren beiden Enden jeweils mit einem Abschlusswiderstand von jeweils ≥ 120 Ω abschließen!

Die Geräte der **ifm electronic gmbh**, die mit einem CAN-Interface ausgestattet sind, haben grundsätzlich keine Abschlusswiderstände.

Zusammen mit den Abschlusswiderständen soll der Gesamtwiderstand (gemessen zwischen CAN_H und CAN_L) der spannungslosen CAN-Buslinie etwa 60±5 Ω betragen.

Stichleitungen

13013

In Abhängigkeit von der Gesamtleitungslänge und den zeitlichen Abläufen auf dem Bus können Signal-Reflektionen auftreten. Daher sollten zu den Busteilnehmern (Node 1 ... Node n) idealerweise keine Stichleitungen führen.

Falls Stichleitungen nicht vermeidbar sind:

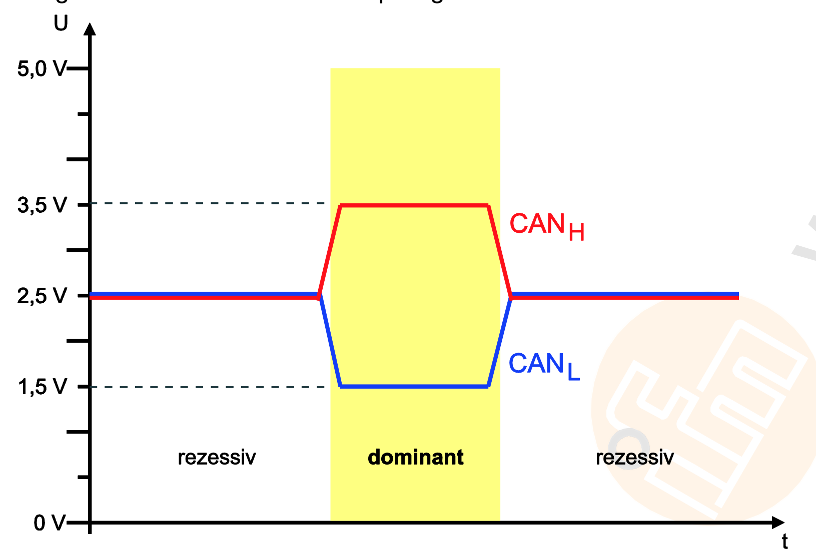
- Einzelne Stichleitungen von bis zu 2 m Länge (bezogen auf 125 kBit/s) gelten als unkritisch.
- Die Summe aller Stichleitungen im Gesamtsystem sollte 30 m nicht übersteigen.

► In besonderen Fällen die Leitungslängen der Linie und der Stiche genau berechnen!

CAN-Buspegel

1179

Der CAN-Bus befindet sich im inaktiven (rezessiven) Zustand, wenn die Ausgangstransistorpaare in allen Busteilnehmern ausgeschaltet sind. Wird mindestens ein Transistorpaar eingeschaltet, wird ein Bit auf den Bus gegeben. Der Bus wird dadurch aktiv (dominant). Es fließt ein Strom durch die Abschlusswiderstände und erzeugt eine Differenzspannung zwischen den beiden Busleitungen. Die rezessiven und dominanten Zustände werden in den Busknoten in entsprechende Spannungen umgewandelt und von den Empfängerschaltkreisen erkannt.



Grafik: Buspegel

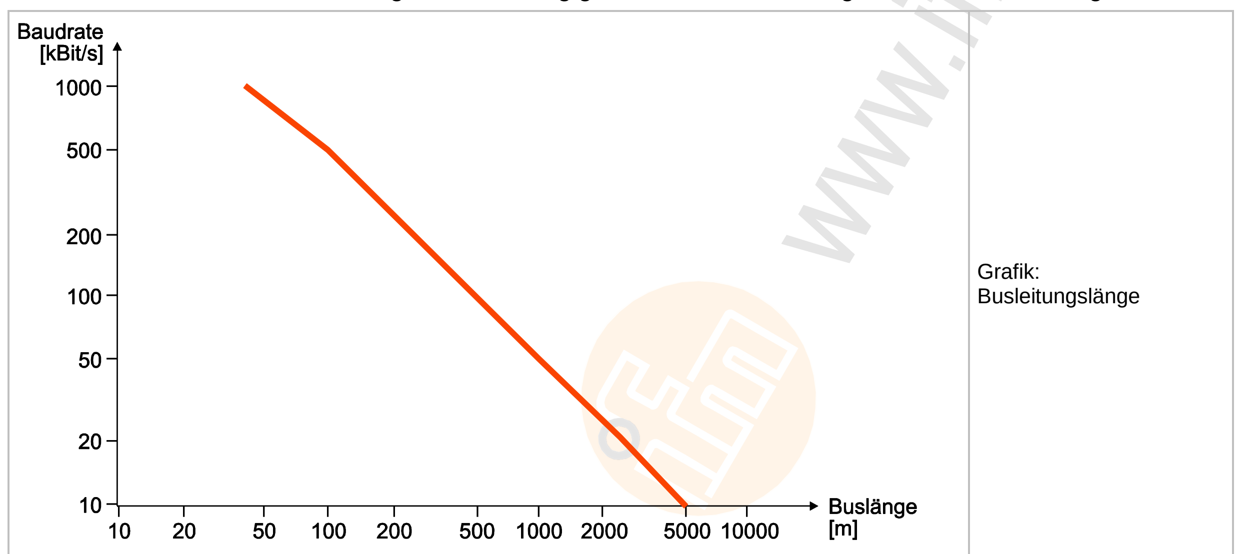
Durch diese differentielle Übertragung mit gemeinsamem Rückleiter wird die Übertragungssicherheit entscheidend verbessert. Störspannungen, die von außen auf das System einwirken, oder Massepotential-Verschiebungen beeinflussen beide Signalleitungen mit gleichen Störgrößen. Dadurch fallen die Störungen bei der Differenzbildung im Empfänger wieder heraus.

Busleitungslänge

1180

Die Länge der Busleitung ist abhängig von:
- Beschaffenheit der Busverbindung (Kabel, Steckverbinder),
- Leitungswiderstand,
- benötigte Übertragungsrate (Baudrate),
- Länge der Stichleitungen.

Vereinfachend kann man von folgender Abhängigkeit zwischen Buslänge und Baudrate ausgehen:



Grafik: Busleitungslänge

Baudrate [kBit/s]	Buslänge [m]	nominelle Bit-Zeit [µs]
1 000	40	1
800	50	1,25
500	100	2
250	250	4
125	500	8
62,5	1 000	20
20	2 500	50
10	5 000	100

Tabelle: Abhängigkeiten Buslänge / Baudrate / Bit-Zeit

Leitungsquerschnitte

1181

► Für die Auslegung des CAN-Netzes auch den Leitungsquerschnitt der eingesetzten Busleitung beachten!

Die folgende Tabelle beschreibt die Abhängigkeit des Leitungsquerschnitts bezogen auf die Leitungslänge und der Anzahl der daran angeschlossenen Teilnehmer (Knoten):

Leitungslänge [m]	Leitungsquerschnitt [mm²]		
	bei 32 Knoten	bei 64 Knoten	bei 100 Knoten
≤ 100	0,25	0,25	0,25
≤ 250	0,34	0,50	0,50
≤ 500	0,75	0,75	1,00

Abhängig von den EMV-Anforderungen können Sie die Busleitungen wie folgt ausführen:
- parallel mit / ohne Abschirmung
- als Twisted-Pair mit / ohne Abschirmung



CAN: Software

Inhalt

14104

IDs (Adressen) in CAN

15448

In CAN werden diverse Arten von Kennungen (hier: IDs) unterschieden:

- COB-ID
  Der **C**ommunication-**Ob**ject-**Id**entifier adressiert die Nachricht (= das Kommunikationsobjekt). Ein Kommunikationsobjekt besteht aus einer netzwerkweiten CAN-Nachricht.
  Je niedriger die COB-ID, desto höher die Priorität der Nachricht.
- Download-ID
  Die Download-ID bezeichnet die Kennung für den Programm-Download und den Wartungszugriff in CODESYS.
  ifm nutzt hierfür den SDO-Mechanismus aus dem CANopen-Protokoll.
  Die Software addiert dazu die Download-ID zur Basisadresse.
- Node-ID
  Der **Node-Id**entifier ist ein eindeutiger Bezeichner für CANopen-Geräte (Devices) im CAN-Netzwerk. Die Node-ID ist auch Bestandteil einiger vordefinierter Verbindungssätze (→ ***Funktions-Code / Predefined Connectionset*** (→ Seite 450)). Anhand der Node-ID werden die COB-IDs ermittelt, wenn die vordefinierten Verbindungseinstellungen verwendet werden.
  ⓘ Die Node-ID und die Download-ID müssen sich im gleichen CAN-Netzwerk unterscheiden!

13182

Gegenüberstellung Download-ID vs. Node-ID

Programm-Download / Wartungszugriff		CANopen	
Download-ID	**COB-ID SDO**	**Node-ID**	**COB-ID SDO**
1…127	TX: 0x580 + Download-ID	1…127	TX: 0x580 + Node-ID
	RX: 0x600 + Download-ID		RX: 0x600 + Node-ID

TX = Slave sendet an Master
RX = Slave empfängt von Master

HINWEIS

Die CAN-Download-ID des Geräts muss mit der in CODESYS eingestellten CAN-Download-ID übereinstimmen!

Im selben CAN-Netzwerk müssen die CAN-Download-IDs einmalig sein!

Den verschiedenen CAN-Schnittstellen eines Geräts darf die gleiche CAN-Download-ID zugewiesen werden, vorausgesetzt, diese CAN-Schnittstellen sind an getrennten CAN-Netzwerken angeschlossen.

COB-ID

18384
18379

Je nach Typ sind folgende COB-Identifier frei verfügbar für den Datentransfer:

COB-ID (base)	COB-ID (extended)
11 Bit	29 Bit
COB-Identifier: 0...2 047	COB-Identifier: 0...536 870 911
Standard-Anwendungen	Motor-Management (SAE J1939), Truck & Trailer Interface (ISO 11992)

18382

Beispiel 11-Bit COB-ID (base):

S O F	COB-ID (base) Bit 28 ... Bit 18											R T R	I D E
0	0	0	0	0	1	1	1	1	1	1	1	0	0
	0			7				F					

Beispiel 29-Bit COB-ID (extended):

S O F	COB-ID (base) Bit 28 ... Bit 18											S R R	I D E	COB-ID (extended) Bit 17 ... Bit 0																		R T R
0	0	0	0	0	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	0	1				F				C						0				0				0				0				

Legende:
SOF = Start of frame
Flanke von rezessiv zu dominant

RTR = **R**emote **t**ransmission **r**equest
dominant: Diese Nachricht liefert Daten
rezessiv: Diese Nachricht fordert Daten an

IDE = **Id**entifier **e**xtension flag
dominant: Hiernach folgen Steuerungs-Bits
rezessiv: Hiernach folgt der zweite Teil des 29-Bit-Identifier

SRR = **S**ubstitute **r**emote **r**equest
rezessiv: Extended CAN-ID: Ersetzt das RTR-Bit an dieser Stelle

4.4 Software

Inhalt

14107

4.4.1 Software-Module für das Gerät

Inhalt

14110

Die Software in diesem Gerät setzt wie folgt auf der Hardware auf:

Software-Modul	Anwender kann das Modul ändern?	womit?
Anwendungsprogramm mit Bibliotheken	ja	CODESYS, MaintenanceTool
Laufzeitsystem (LZS) *)	Upgrade ja Downgrade ja	MaintenanceTool
Bootloader	nein	---
(Hardware)	nein	---

*) Die Laufzeitsystem-Versionsnummer muss der Target-Versionsnummer in der CODESYS-Zielsystemeinstellung entsprechen!
→ Kapitel ***Target einrichten*** (→ Seite 173)

Nachfolgend beschreiben wir diese Software-Module:

Bootloader

14111

Im Auslieferungszustand enthalten **ecomatmobile**-Controller nur den Bootloader.
Der Bootloader ist ein Startprogramm, mit dem das Laufzeitsystem und das Anwendungsprogramm auf dem Gerät nachgeladen werden können.
Der Bootloader enthält Grundroutinen...

- zur Kommunikation der Hardware-Module untereinander,
- zum Nachladen des Laufzeitsystems.

Der Bootloader ist das erste Software-Modul, das im Gerät gespeichert sein muss.

Laufzeitsystem

14112

Grundprogramm im Gerät, stellt die Verbindung her zwischen der Hardware des Gerätes und dem Anwendungsprogramm.
→ Kapitel ***Software-Module für das Gerät*** (→ Seite 159)

Im Auslieferungszustand ist im Normalfall kein Laufzeitsystem im Controller geladen (LED blinkt grün mit 5 Hz). In diesem Betriebszustand ist nur der Bootloader aktiv. Dieser stellt die minimalen Funktionen für den Laufzeitsystem-Ladevorgang zur Verfügung, u.a. die Unterstützung der Schnittstellen (z.B. CAN).

Der Laufzeitsystem-Download muss im Normalfall nur einmalig durchgeführt werden. Das Anwendungsprogramm kann anschließend (auch mehrmals) in den Controller geladen werden, ohne das Laufzeitsystem zu beeinflussen.

Das Laufzeitsystem wird zusammen mit dieser Dokumentation auf einem separaten Datenträger zur Verfügung gestellt. Zusätzlich kann auch die aktuelle Version von der Homepage der **ifm electronic gmbh** heruntergeladen werden:
→ www.ifm.com > [Service] > [Download]

Anwendungsprogramm

14118

Software, die speziell für die Anwendung vom Hersteller in die Maschine programmiert wird. Die Software enthält üblicherweise logische Sequenzen, Grenzwerte und Ausdrücke zum Steuern der entsprechenden Ein- und Ausgänge, Berechnungen und Entscheidungen.

8340

⚠ WARNUNG

Für die sichere Funktion der Anwendungsprogramme, die vom Anwender erstellt werden, ist dieser selbst verantwortlich. Bei Bedarf muss er zusätzlich entsprechend der nationalen Vorschriften eine Abnahme durch entsprechende Prüf- und Überwachungsorganisationen durchführen lassen.

Bibliotheken

14117

ifm electronic bietet passend für jedes Gerät eine Reihe von Bibliotheken (`*.LIB`) an, die Programmmodule für das Anwendungsprogramm enthalten. Beispiele:

Bibliothek	Verwendung
`ifm_CR7132_Vxxyyzz.LIB`	gerätespezifische Bibliothek Muss immer im Anwendungsprogramm enthalten sein!
`ifm_CR7132_CANopenxMaster_Vxxyyzz.LIB` x = 1...4 = Nummer der CAN-Schnittstelle	(optional) wenn eine CAN-Schnittstelle des Geräts als CANopen-Master betrieben werden soll
`ifm_CR7132_CANopenxSlave_Vxxyyzz.LIB` x = 1...4 = Nummer der CAN-Schnittstelle	(optional) wenn eine CAN-Schnittstelle des Geräts als CANopen-Slave betrieben werden soll
`ifm_CR7132_J1939_Vxxyyzz.LIB`	(optional) wenn eine CAN-Schnittstelle des Geräts mit einer Motorsteuerung kommunizieren soll

→ Kapitel ***ifm-Bibliotheken für das Gerät CR7132*** (→ Seite 190)

4.4.2 Programmierhinweise für CODESYS-Projekte

Inhalt

7426

Hier erhalten Sie Tipps zum Programmieren des Geräts.

► Beachten Sie die Hinweise im CODESYS-Programmierhandbuch
→ **ecomat*mobile***-DVD "Software, tools and documentation".

Angaben zur Software

14466

Wir beziehen uns in dieser Anleitung auf CODESYS ab Version 2.3.9.42 (jedoch NICHT Version 3).

> ⓘ Für sicherheitsgerichtete Anwendungen nur eine von der **ifm electronic** dafür zur Verfügung gestellte CODESYS-Version verwenden!
> Fragen Sie Ihren **ecomat*mobile***-Fachberater!

Im "Programmierhandbuch CODESYS 2.3" erhalten Sie weitergehende Informationen über die Nutzung des Programmiersystems "CODESYS for Automation Alliance". Dieses Handbuch steht auf der **ifm**-Internetseite als kostenloser Download zur Verfügung:
→ www.ifm.com > [Service] > [Download]
→ **ecomat*mobile***-DVD "Software, tools and documentation"

7649

Das Anwendungsprogramm nach IEC 61131-3 kann vom Anwender komfortabel mit dem Programmiersystem CODESYS selbst erstellt werden. Für den Einsatz dieser Software auf dem PC gelten unter anderem folgende Systemvoraussetzungen:

- Betriebssysteme (unter anderem):
  - Windows Vista (32 Bit) (ⓘ nicht getestet, ohne Gewähr für **ifm**-Produkte)
  - Windows 7 (32/64 Bit) ab CODESYS 2.3.9.26

Weitere Details zur aktuellen CODESYS-Software → www.codesys.com

► Der Anwender muss außerdem beachten, welcher Softwarestand (speziell beim R360-Laufzeitsystem und den Funktionsbibliotheken) zum Einsatz kommt.

2689

HINWEIS

Es müssen immer die zum gewählten Target passenden Software-Stände zum Einsatz kommen:
• des Laufzeitsystems (`ifm_CR7132_Vxxyyzz.H86`),
• der Steuerungskonfiguration (`ifm_CR7132_Vxx.CFG`),
• der Gerätebibliothek (`ifm_CR7132_Vxxyyzz.LIB`) und
• der weiteren Dateien

V	Version
xx: 00...99	Versionsnummer
yy: 00...99	Release-Nummer
zz: 00...99	Patch-Nummer

Dabei müssen der Basisdateiname (z.B. "`CR7132`") und die Software-Versionsnummer "xx" (z.B. "`02`") überall den gleichen Wert haben! Andernfalls geht das Gerät in den STOP-Zustand.
Die Werte für "yy" (Release-Nummer) und "zz" (Patch-Nummer) müssen **nicht** übereinstimmen.

4368

Folgende Dateien müssen ebenfalls geladen sein:
• die zum Projekt erforderlichen internen Bibliotheken (in IEC 61131 erstellt),
• die Konfigurationsdateien (`*.CFG`)
• und die Target-Dateien (`*.TRG`).

Es kann vorkommen, dass das Zielsystem mit Ihrer aktuell installierten Version von CODESYS nicht oder nur teilweise programmiert werden kann. Im diesem Fall wenden Sie sich bitte an den technischen Support der **ifm electronic gmbh**.

Software-Tool Downloader: → Kapitel ***ifm-Downloader nutzen*** (→ Seite 167)

8340

⚠ WARNUNG

Für die sichere Funktion der Anwendungsprogramme, die vom Anwender erstellt werden, ist dieser selbst verantwortlich. Bei Bedarf muss er zusätzlich entsprechend der nationalen Vorschriften eine Abnahme durch entsprechende Prüf- und Überwachungsorganisationen durchführen lassen.

FB, FUN, PRG in CODESYS

8473

In CODESYS unterscheiden wir folgende Typen von Bausteinen (POUs):

FB = function block = Funktionsbaustein

- Ein FB kann mehrere Eingänge und mehrere Ausgänge haben.
- Ein FB darf in einem Projekt mehrmals aufgerufen werden.
- Für jeden Aufruf muss eine Instanz deklariert werden.
- Erlaubt: Im FB aufrufen von FB und FUN.

FUN = function = Funktion

- Eine Funktion kann mehrere Eingänge, aber nur einen Ausgang haben.
- Der Ausgang ist vom gleichen Datentyp wie die Funktion selbst.

PRG = program = Programm

- Ein PRG kann mehrere Eingänge und mehrere Ausgänge haben.
- Ein PRG darf in einem Projekt nur einmal aufgerufen werden.
- Erlaubt: im PRG aufrufen von PRG, FB und FUN.

HINWEIS

Funktionsbausteine dürfen NICHT in Funktionen aufgerufen werden!
Sonst: Bei der Ausführung stürzt das Anwendungsprogramm ab.

Alle Bausteine (POUs) dürfen NICHT rekursiv aufgerufen werden, auch nicht indirekt!

Eine IEC-Anwendung darf maximal 8000 Bausteine (POU) enthalten!

Hintergrund:

Alle Variablen von Funktionen...

- werden beim Aufruf initialisiert und
- werden nach der Rückkehr zum Aufrufer ungültig.

Funktionsbausteine haben 2 Aufrufe:

- einen Initialisierungsaufruf und
- den eigentlichen Aufruf, um irgend etwas zu tun.

Folglich heißt das für den FB-Aufruf in einer Funktion:

- jedesmal erfolgt ein zusätzlicher Initialisierungsaufruf und
- die Daten des letzten Aufrufs gehen verloren.

Berechnungen und Konvertierungen im Anwendungsprogramm

16356

Ab Laufzeitsystem V01.00.04 gilt:

HINWEIS

Falls folgende Elemente im Anwendungsprogramm erforderlich sind:
• mathematische Funktionen (z.B. ATAN),
• Berechnungen,
• Konvertierungen (z.B. REAL_TO_BYTE),
dann gilt für die Werte an den Eingängen und Ausgängen der entsprechenden Operatoren:

► Den zulässigen Wertebereich in jedem Einzelfall unbedingt einhalten!

> Ansonsten kann es zu einem fatalen Fehler in der Steuerung kommen.

Zykluszeit beachten!

8006

Bei den frei programmierbaren Geräten aus der Controller-Familie **ecomat*mobile*** stehen in einem großen Umfang Bausteine zur Verfügung, die den Einsatz der Geräte in den unterschiedlichsten Anwendungen ermöglichen.

Da diese Bausteine je nach Komplexität mehr oder weniger Systemressourcen belegen, können nicht immer alle Bausteine gleichzeitig und mehrfach eingesetzt werden.

ACHTUNG

Gefahr von zu trägem Verhalten des Geräts!
Zykluszeit darf nicht zu lang werden!

► Beim Erstellen des Anwendungsprogramms die oben aufgeführten Empfehlungen beachten und durch Austesten überprüfen.

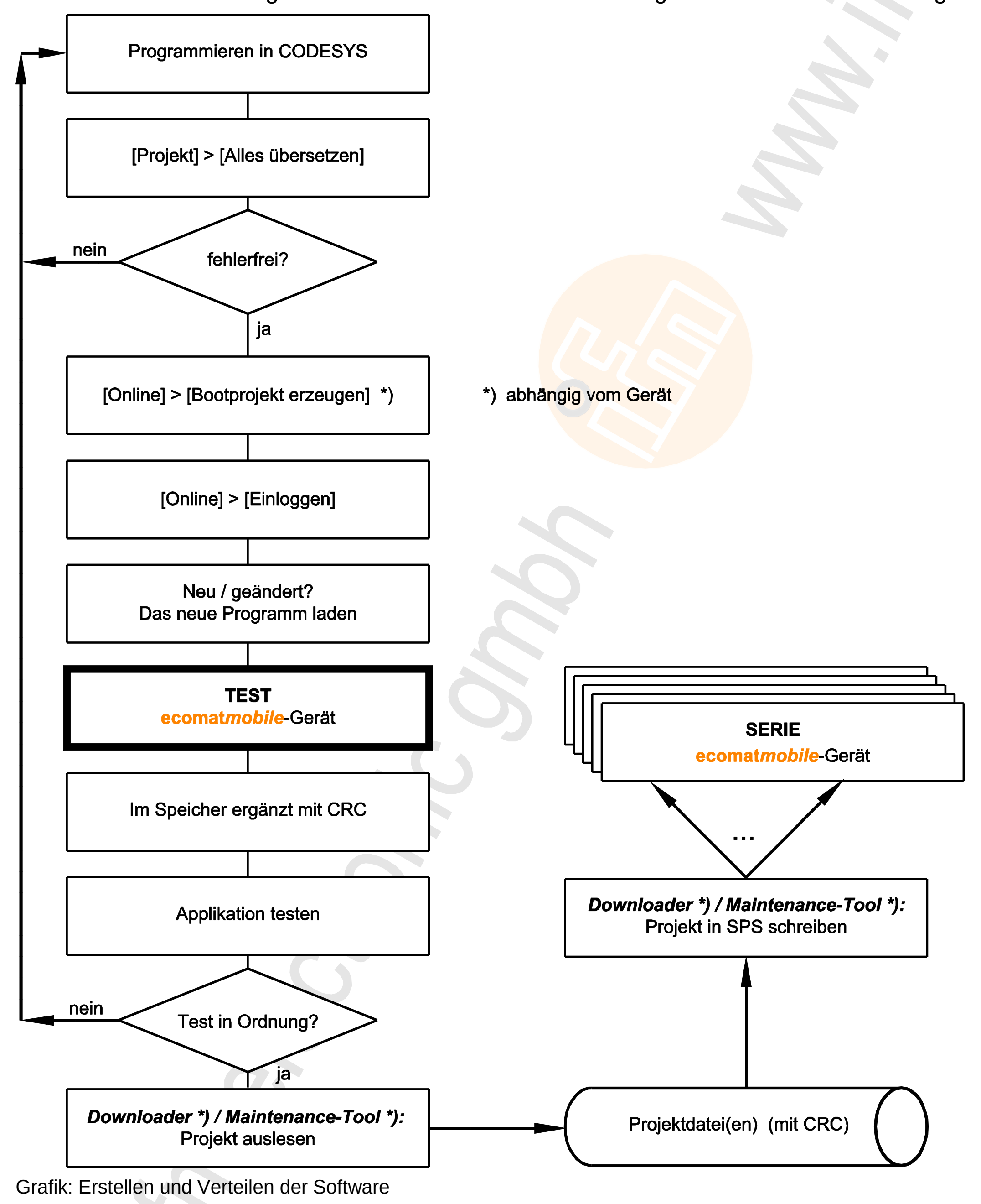
► Bei Bedarf durch Neustrukturieren der Software und des Systemaufbaus die Zykluszeit vermindern.

Anwendungsprogramm erstellen

8007

Das Anwendungsprogramm wird mit dem Programmiersystem CODESYS erstellt und während der Programmentwicklung mehrfach zum Testen in die Steuerung geladen:
In CODESYS: [Online] > [Einloggen] > das neue Programm laden.

Für jeden derartigen Download via CODESYS wird dazu der Quellcode neu übersetzt. Daraus resultiert, dass auch jedes Mal im Speicher der Steuerung eine neue Prüfsumme gebildet wird. Auch für Sicherheitssteuerungen ist dieses Verfahren bis zur Freigabe der Software zulässig.



Grafik: Erstellen und Verteilen der Software

Boot-Projekt speichern

7430

Speichern Sie im Gerät zusammen mit Ihrem Anwendungsprogramm immer auch das zugehörige Boot-Projekt! Nur so ist das Anwendungsprogramm auch nach einem Spannungsausfall im Gerät verfügbar.

HINWEIS

Beachten: das Boot-Projekt ist etwas größer als das eigentliche Programm.

Jedoch: das Speichern des Boot-Projekts im Gerät wird scheitern, wenn das Boot-Projekt größer wird als der vorhandene IEC-Code-Speicherbereich. Nach Power-On-Reset ist das Boot-Projekt wieder gelöscht oder ungültig.

► CODESYS-Menü [Online] > [Bootprojekt erzeugen]
Dies muss auch nach jeder Änderung erneut erfolgen!

> Nach einem Neustart startet das Gerät mit dem zuletzt gespeicherten Boot-Projekt.

> Falls noch KEIN Boot-Projekt gespeichert wurde:
• das Gerät bleibt nach dem Neustart im STOP-Betrieb
• das Anwendungsprogramm ist nicht (mehr) vorhanden
• die LED leuchtet grün.

ifm-Downloader nutzen

8008

Der **ifm**-Downloader dient dem einfachen Übertragen des Programmcodes vom Programmierplatz in die Steuerung. Grundsätzlich kann jedes Anwendungsprogramm mit dem **ifm**-Downloader auf die Steuerungen kopiert werden. Vorteil: Dazu ist kein Programmiersystem mit einer CODESYS-Lizenz erforderlich.

Hier finden Sie den aktuellen **ifm**-Downloader (min. V06.18.26):
ecomat*mobile*-DVD "Software, tools and documentation" im Register "R360 tools [D/E]"

Umgang mit sicherheitsrelevanter Software

8005

► Ein sicherheitsrelevantes Anwendungsprogramm NUR mit dem **ifm**-Downloader auf die Steuerungen kopieren!
Anders kann nicht gewährleistet werden, dass die zertifizierte Software ohne Änderungen vervielfältigt wird.

→ Kapitel ***Handhabung von sicherheitsrelevanter Software*** (→ Seite 119)
(im Kapitel ***Hinweise für sicherheitsrelevante Anwendungen*** (→ Seite 15))

4.4.3 Betriebszustände

13980

→ Kapitel ***Betriebszustände*** (→ Seite 39)
(im Kapitel ***Hinweise für sicherheitsrelevante Anwendungen*** (→ Seite 15))

4.4.4 Leistungsgrenzen des Geräts

7358



Leistungsgrenzen des Geräts beachten! → Datenblatt

Maximale Programmlaufzeit

11791

Die maximale Programmlaufzeit beträgt 100 ms.
→ ***Programmablauf und Zykluszeit überwachen*** (→ Seite 45)
→ ***Watchdog*** (→ Seite 487) (Glossary)

CODESYS-Funktionen

2254

Folgende Grenzen sollten Sie berücksichtigen:

- Bis zu 2048 Bausteine (PB, FB...) werden unterstützt.
- Für Merker stehen 8 kByte zur Verfügung, davon 4 kByte für remanente Merker (Retain). Beschreibung der Retain-Merker → bei den jeweiligen FBs.

5 Konfigurationen

Inhalt

Die in den jeweiligen Montage- und Installationsanweisungen oder dem ***Anhang*** (→ Seite 420) dieser Dokumentation beschriebenen Gerätekonfigurationen stehen als Standardgeräte (Lagerware) zur Verfügung. Diese decken bei den meisten Anwendungen die geforderten Spezifikationen ab.

Entsprechend den Kundenanforderungen bei Serieneinsatz ist es aber auch möglich, dass andere Gerätekonfigurationen z.B. hinsichtlich der Zusammenstellung der Ein- und Ausgänge und der Ausführung der Analogkanäle eingesetzt werden.

5.1 Laufzeitsystem einrichten

Inhalt

14091

5.1.1 Laufzeitsystem neu installieren

14177
2733

Im Auslieferungszustand ist im Normalfall kein Laufzeitsystem im Gerät geladen (LED blinkt grün mit 5 Hz). In diesem Betriebszustand ist nur der Bootloader aktiv. Dieser stellt die minimalen Funktionen für den Laufzeitsystem-Ladevorgang zur Verfügung, u.a. die Unterstützung der Schnittstellen (z.B. RS232, CAN).

Der Laufzeitsystem-Download muss im Normalfall nur einmalig durchgeführt werden. Das Anwendungsprogramm kann anschließend (auch mehrmals) in das Gerät geladen werden, ohne das Laufzeitsystem zu beeinflussen.

Das Laufzeitsystem wird zusammen mit dieser Dokumentation auf einem separaten Datenträger zur Verfügung gestellt. Zusätzlich kann auch die aktuelle Version von der Homepage der **ifm electronic gmbh** heruntergeladen werden:
→ www.ifm.com > [Service] > [Download]

2689

HINWEIS

Es müssen immer die zum gewählten Target passenden Software-Stände zum Einsatz kommen:
- des Laufzeitsystems (`ifm_CR7132_Vxxyyzz.H86`),
- der Steuerungskonfiguration (`ifm_CR7132_Vxx.CFG`),
- der Gerätebibliothek (`ifm_CR7132_Vxxyyzz.LIB`) und
- der weiteren Dateien

V	Version
xx: 00...99	Versionsnummer
yy: 00...99	Release-Nummer
zz: 00...99	Patch-Nummer

Dabei müssen der Basisdateiname (z.B. "`CR7132`") und die Software-Versionsnummer "xx" (z.B. "`02`") überall den gleichen Wert haben! Andernfalls geht das Gerät in den STOP-Zustand.

Die Werte für "yy" (Release-Nummer) und "zz" (Patch-Nummer) müssen **nicht** übereinstimmen.

4368

Folgende Dateien müssen ebenfalls geladen sein:
- die zum Projekt erforderlichen internen Bibliotheken (in IEC 61131 erstellt),
- die Konfigurationsdateien (`*.CFG`)
- und die Target-Dateien (`*.TRG`).

Es kann vorkommen, dass das Zielsystem mit Ihrer aktuell installierten Version von CODESYS nicht oder nur teilweise programmiert werden kann. Im diesem Fall wenden Sie sich bitte an den technischen Support der **ifm electronic gmbh**.

Das Laufzeitsystem wird mit dem eigenständigen Programm "**ifm**-Downloader" in das Gerät übertragen. (Der **ifm**-Downloader und dessen Dokumentation befindet sich auf der **ecomat*mobile***-DVD "Software, tools and documentation" oder kann bei Bedarf von der **ifm**-Homepage heruntergeladen werden: → www.ifm.com > [Service] > [Download]).

Das Anwendungsprogramm wird im Normalfall über das Programmiersystem in das Gerät geladen. Es kann aber ebenfalls mit dem **ifm**-Downloader geladen werden, wenn es zuvor aus dem Gerät ausgelesen wurde (→ Upload).

5.1.2 Installation verifizieren

14178

Das sicherheitsrelevante Anwendungsprogramm ist nur zertifizierungsfähig zusammen mit einem als zugehörig definierten Laufzeitsystem.

► Nach dem Laden des Laufzeitsystems in die Steuerung:
 • Prüfen, ob das Laufzeitsystem korrekt übertragen wurde!
 • Prüfen, ob sich das richtige Laufzeitsystem auf der Steuerung befindet!

► 1. Prüfung:
mit dem **ifm**-Downloader prüfen, ob die richtige Laufzeitsystem-Version geladen wurde:
 • Name, Version und die CRC des Laufzeitsystems im Gerät auslesen!
 • Diese Daten manuell mit den Soll-Daten vergleichen!

► 2. Prüfung:
Im Anwendungsprogramm prüfen, ob die richtige Laufzeitsystem-Version geladen wurde:
 • Name und die Version des Laufzeitsystems im Gerät auslesen!
 • Diese Daten mit fest vorgegebenen Werten vergleichen!
Zum Auslesen der Daten dient folgender FB:

GET_IDENTITY (→ Seite 377)	liest die im Gerät gespeicherten spezifischen Kennungen: • Hardware-Name und Hardware-Version des Geräts • Seriennummer des Geräts • Name des Laufzeitsystems im Gerät • Version und Ausgabe des Laufzeitsystems im Gerät • Name der Anwendung (wurde zuvor mit ***SET_IDENTITY*** (→ Seite 379) gespeichert)

► Wird durch die Anwendung eine falsche Laufzeitsystem-Version erkannt:
alle Sicherheitsfunktionen in den sicheren Zustand schalten!
ℹ Sicherer Zustand = Das Anwendungsprogramm deaktiviert alle entsprechend programmierten Ausgänge.
Falls ein anwendungsspezifischer Fehler mit dem FB ***ERROR_REPORT*** (→ Seite 382) an das Laufzeitsystem gemeldet wurde:
Das Laufzeitsystem deaktiviert alle sicheren Ausgänge und stoppt das Senden von CANsafety-Nachrichten. **Nicht** deaktiviert werden die für diesen Fehler mit ***SET_KEEP_ALIVE*** (→ Seite 386) ausgenommenen Ausgänge und CANsafety-Schnittstellen.

5.2 Programmiersystem einrichten

Inhalt

3968

5.2.1 Programmiersystem manuell einrichten

Inhalt

3963

Target einrichten

2687
11379

Beim Erstellen eines neuen Projektes in CODESYS muss die dem Gerät entsprechende Target-Datei geladen werden.

► Im Dialog-Fenster [Zielsystem Einstellungen] im Menü [Konfiguration] die gewünschte Target-Datei wählen.

> Die Target-Datei stellt für das Programmiersystem die Schnittstelle zur Hardware her.

> Gleichzeitig mit Wahl des Targets werden automatisch einige wichtige Bibliotheken und die Steuerungskonfiguration geladen.

► Bei Bedarf im Fenster [Zielsystem Einstellungen] > Reiter [Netzfunktionen] > [Parameter-Manager unterstützen] und / oder [Netzvariablen unterstützen] aktivieren.

► Bei Bedarf geladene (3S-)Bibliotheken wieder entfernen oder durch weitere (ifm-)Bibliotheken ergänzen.

► Immer die passende Geräte-Bibliothek `ifm_CR7132_Vxxyyzz.LIB` manuell ergänzen!

2689

HINWEIS

Es müssen immer die zum gewählten Target passenden Software-Stände zum Einsatz kommen:
- des Laufzeitsystems (`ifm_CR7132_Vxxyyzz.H86`),
- der Steuerungskonfiguration (`ifm_CR7132_Vxx.CFG`),
- der Gerätebibliothek (`ifm_CR7132_Vxxyyzz.LIB`) und
- der weiteren Dateien

V	Version
xx: 00...99	Versionsnummer
yy: 00...99	Release-Nummer
zz: 00...99	Patch-Nummer

Dabei müssen der Basisdateiname (z.B. "`CR7132`") und die Software-Versionsnummer "xx" (z.B. "`02`") überall den gleichen Wert haben! Andernfalls geht das Gerät in den STOP-Zustand.

Die Werte für "yy" (Release-Nummer) und "zz" (Patch-Nummer) müssen **nicht** übereinstimmen.

4368

Folgende Dateien müssen ebenfalls geladen sein:
- die zum Projekt erforderlichen internen Bibliotheken (in IEC 61131 erstellt),
- die Konfigurationsdateien (`*.CFG`)
- und die Target-Dateien (`*.TRG`).

Es kann vorkommen, dass das Zielsystem mit Ihrer aktuell installierten Version von CODESYS nicht oder nur teilweise programmiert werden kann. Im diesem Fall wenden Sie sich bitte an den technischen Support der **ifm electronic gmbh**.

Steuerungskonfiguration aktivieren (z.B. CR0033)

15824

Bei der Konfiguration des Programmiersystems (→ vorheriger Abschnitt) erfolgte automatisch auch die Steuerungskonfiguration.

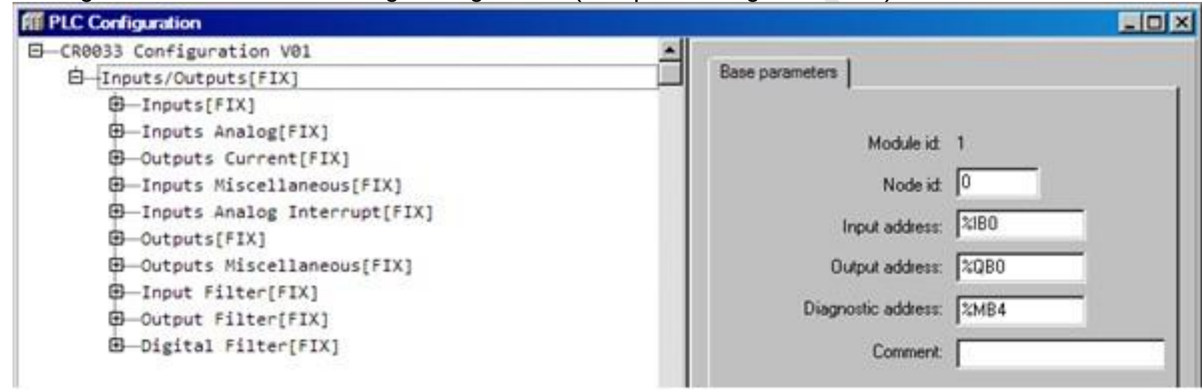
► Den Punkt [Steuerungskonfiguration] erreicht man über den Reiter [Ressourcen].
Mit Doppelklick auf den Punkt [Steuerungskonfiguration] öffnet sich das entsprechende Fenster.

► In CODESYS den Reiter [Ressourcen] klicken:



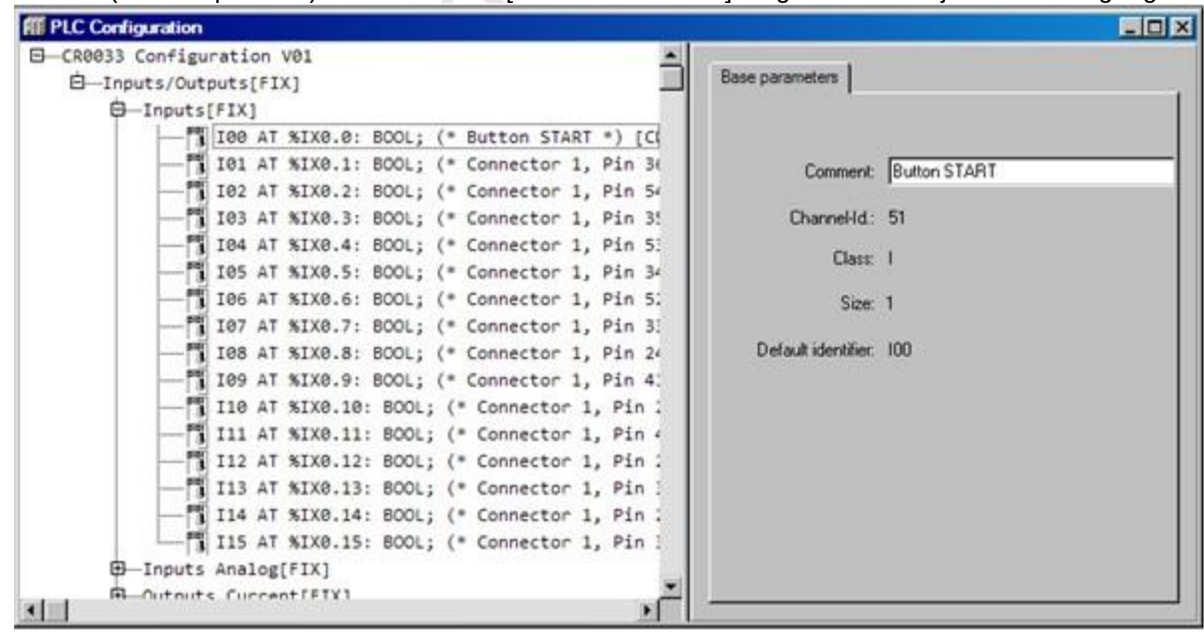
► In der linken Spalte Doppelklick auf [Steuerungskonfiguration]

> Anzeige der aktuellen Steuerungskonfiguration (Beispiel → folgendes Bild):



Durch die Konfiguration ist für den Anwender in der Programmumgebung Folgendes verfügbar:

- alle wichtigen System- und Fehlermerker
  Je nach Anwendung und Anwendungsprogramm müssen diese Merker bearbeitet und ausgewertet werden. Der Zugriff erfolgt über deren symbolischen Namen.
- die Struktur der Ein- und Ausgänge
  Diese können im Fenster [Steuerungskonfiguration] (→ Bild unten) direkt symbolisch bezeichnet werden (sehr empfohlen!) und stehen als [Globale Variablen] im gesamten Projekt zur Verfügung.



5.2.2 Programmiersystem über Templates einrichten

13745

ifm bietet vorgefertigte Templates (Programm-Vorlagen), womit Sie das Programmiersystem schnell, einfach und vollständig einrichten können.

970

Beim Installieren der **ecomat*mobile***-DVD "Software, tools and documentation" wurden auch Projekte mit Vorlagen auf Ihrem Computer im Programmverzeichnis abgelegt:
`…\ifm electronic\CoDeSys V…\Projects\Template_DVD_V…`

► Die gewünschte dort gespeicherte Vorlage in CODESYS öffnen mit:
[Datei] > [Neu aus Vorlage…]

> CODESYS legt ein neues Projekt an, dem der prinzipielle Programmaufbau entnommen werden kann. Es wird dringend empfohlen, dem gezeigten Schema zu folgen.

5.3 Funktionskonfiguration, allgemein

Inhalt

3971

5.3.1 Konfiguration der Ein- und Ausgänge (Voreinstellung)

2249

- Alle Ein-/Ausgänge sind im Auslieferungszustand im Binär-Modus (plus-schaltend!).
- Die Diagnosefunktion ist nicht aktiv.
- Der Überlastschutz ist aktiv.

5.3.2 Systemvariablen

13519
15576

Alle Systemvariablen (→ Kapitel ***Systemmerker*** (→ Seite 420)) liegen auf festen, nicht verschiebbaren Adressen.

> Zur Anzeige und Verarbeitung eines Watchdog-Fehlers oder Ursachen eines Neustarts wird die Systemvariable LAST_RESET gesetzt.

5.4 Funktionskonfiguration der Ein- und Ausgänge

Inhalt

1812
1394

Bei bestimmten Ein- und Ausgängen sind zusätzliche Diagnosefunktionen aktivierbar. Damit kann das jeweilige Ein- und Ausgangssignal überwacht werden und im Fehlerfall kann das Anwendungsprogramm darauf reagieren.

Je nach Ein- und Ausgang müssen bei der Nutzung der Diagnose bestimmte Randbedingungen beachtet werden:

► Anhand des Datenblattes prüfen, für welche Ein- und Ausgänge des Geräts welche Diagnosemöglichkeit zur Verfügung steht!

- Zur Konfiguration der Ein- und Ausgänge sind in den Gerätebibliotheken (`ifm_CR7132_Vxxyyzz.LIB`) Konstanten vordefiniert (z.B. IN_DIGITAL_H).
  Ausführliche Angaben → Kapitel ***Mögliche Betriebsarten Ein-/Ausgänge*** (→ Seite 436).

Nur ExtendedController:
Die Namen der Ein- und Ausgänge in der zweiten Steuerungshälfte werden durch ein angehängtes "_E" gekennzeichnet.

5.4.1 Eingänge konfigurieren

Inhalt

3973

Zulässige Betriebsarten → Kapitel ***Mögliche Betriebsarten Ein-/Ausgänge*** (→ Seite 436)

Sicherheitshinweise zu Reed-Relais

7348

Beim Einsatz von nichtelektronischen Schaltern Folgendes beachten:

> ⓘ Kontakte von Reed-Relais können (reversibel) verkleben, wenn sie ohne Vorwiderstand an den Geräte-Eingängen angeschlossen werden.

► **Abhilfe:** Vorwiderstand zum Reed-Relais installieren:
Vorwiderstand = max. Eingangsspannung / zulässiger Strom im Reed-Relais
Beispiel: 32 V / 500 mA = 64 Ohm

► Der Vorwiderstand darf 5 % des Eingangswiderstands RE des Geräte-Eingangs (→ Datenblatt) nicht überschreiten. Sonst wird das Signal nicht als TRUE erkannt.
Beispiel:
RE = 3 000 Ohm
⇨ max. Vorwiderstand = 150 Ohm

Software-Filter der Eingänge konfigurieren

13941
6883

Über die Systemvariablen Ixx_FILTER kann ein Software-Filter konfiguriert werden, der die gemessene Eingangsspannung an den Analogeingängen filtert. Der Filter verhält sich wie ein klassischer Tiefpassfilter, wobei die Grenzfrequenz durch den in die Systemvariable eingetragenen Wert eingestellt wird. Es sind Werte von 0...8 möglich.

Tabelle: Grenzfrequenz Software-Tiefpassfilter am Analogeingang

Ixx_FILTER	Filterfrequenz [Hz]	Signalanstiegszeit	Hinweise
0	Filter deaktiviert		
1	390	1 ms	
2	145	2,5 ms	
3	68	5 ms	
4	34	10 ms	empfohlen, Voreinstellung
5	17	21 ms	
6	8	42 ms	
7	4	84 ms	
8	2	169 ms	
≥ 9	34	10 ms	→ Voreinstellung

(!) Nach dem Ändern der Filtereinstellung wird der Wert dieses Ein- oder Ausgangs nicht sofort richtig ausgegeben. Erst nach der Signalanstiegszeit (→ Tabelle) ist der Wert wieder korrekt.

(i) Die Signalanstiegszeit ist die Zeitdauer, die ein Signal am Ausgang des Filters benötigt, um von 10 % auf 90 % des Endwerts zu kommen, wenn am Eingang ein Sprung angelegt wird. Die Signalabstiegszeit ist die Zeitdauer von 90 % bis 10 %.

(!) Sichere Eingänge dürfen nur mit Software-Filterstufe 4 betrieben werden!

Analogeingänge: Konfiguration und Diagnose

13960

► Die Konfiguration jedes einzelnen Eingangs erfolgt über das Anwendungsprogramm:
• FB ***INPUT_ANALOG*** (→ Seite 261) > Eingang MODE oder:
• FB ***SET_INPUT_MODE*** (→ Seite 264) > Eingang MODE

> Werden die Analogeingänge auf Strommessung konfiguriert, wird bei Überschreiten des Endwertes (21,7 mA) in den sicheren Spannungsmessbereich (0...32 V DC) geschaltet und das jeweilige Fehlerbit im Merkerbyte ERROR_CURRENT_Ix gesetzt. Sinkt der Wert wieder unter den Grenzwert, schaltet der Eingang selbsttätig auf den Strommessbereich zurück.

> Im Anwendungsprogramm können die Systemvariablen ANALOG00...ANALOGxx zur kundenspezifischen Diagnose der Eingänge dienen.

18414

Falls Eingang I15 nicht verwendet:
► Eingang I15 als Binäreingang konfigurieren!

Eingangssignale filtern

13883

→ Kapitel ***Software-Filter der Eingänge konfigurieren*** (→ Seite 179)

Sichere Eingänge

12249

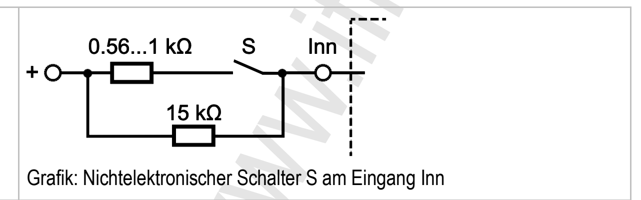
→ Kapitel ***Eingänge für Sicherheitsfunktionen*** (→ Seite 55)
(im Kapitel ***Hinweise für sicherheitsrelevante Anwendungen*** (→ Seite 15))

Binäreingänge: Konfiguration und Diagnose

13962

► Die Konfiguration jedes einzelnen Eingangs erfolgt über das Anwendungsprogramm:
 • FB ***INPUT_ANALOG*** (→ Seite 261) > Eingang MODE oder:
 • FB ***SET_INPUT_MODE*** (→ Seite 264) > Eingang MODE

► Soll die Diagnose genutzt werden, dann diesen Modus zusätzlich aktivieren:
 • FB SET_INPUT_MODE > Eingang DIAGNOSTICS setzen.

Diagnose bei nichtelektronischen Schaltern: ► Schalter mit einer zusätzlichen Widerstandsbeschaltung versehen! ❗ Für sichere Anwendungen ist das Beispiel NICHT geeignet.	 Grafik: Nichtelektronischer Schalter S am Eingang Inn

> Das Diagnose-Ergebnis zeigen z.B. folgende Systemmerker:
 • ERROR_SHORT_Ix meldet einen Kurzschluss an der Eingangsgruppe x
 • ERROR_BREAK_Ix meldet einen Leiterbruch an der Eingangsgruppe x

> Im Anwendungsprogramm können die Systemvariablen ANALOG00...ANALOGxx zur kundenspezifischen Diagnose der Eingänge dienen.

Eingangssignale filtern

13883

→ Kapitel ***Software-Filter der Eingänge konfigurieren*** (→ Seite 179)

Sichere Eingänge

12249

→ Kapitel ***Eingänge für Sicherheitsfunktionen*** (→ Seite 55)
(im Kapitel ***Hinweise für sicherheitsrelevante Anwendungen*** (→ Seite 15))

Schnelle Eingänge

2193

Die Geräte verfügen über schnelle Zähl-/Impulseingänge für eine Eingangsfrequenz bis 30 kHz (→ Datenblatt).

Der Eingangswiderstand der schnellen Eingänge schaltet automatisch um, je nach verwendetem Modus oder Funktionsblock:

Eingangswiderstand	bei Modus / FB
3,2 kOhm	(Standard) FAST_COUNT, FREQUENCY, INC_ENCODER, PERIOD und ähnliche FBs
50,7 kOhm	Messeingang 32 V

❗ Werden z.B. mechanische Schalter an diesen Eingängen angeschlossen, kann es durch Kontaktprellen zu Fehlsignalen in der Steuerung kommen.

▶ Bei Bedarf diese "Fehlsignale" über die Filter Ixx_DFILTER (→ Kapitel ***Systemmerker*** (→ Seite 420)) ausfiltern (nicht für alle Eingänge verfügbar).

Geeignete Funktionsbausteine sind z.B.:

FAST_COUNT (→ Seite 298)	Zählerbaustein für schnelle Eingangsimpulse
FREQUENCY (→ Seite 300)	misst die Frequenz des am gewählten Kanal ankommenden Signals
FREQUENCY_PERIOD (→ Seite 302)	misst die Frequenz und die Periodendauer (Zykluszeit) in [µs] am angegebenen Kanal
INC_ENCODER (→ Seite 304)	Vorwärts-/Rückwärts-Zählerfunktion zur Auswertung von Drehgebern
PERIOD (→ Seite 306)	misst am angegebenen Kanal die Frequenz und die Periodendauer (Zykluszeit) in [µs]
PERIOD_RATIO (→ Seite 308)	misst die Frequenz und die Periodendauer (Zykluszeit) in [µs] über die angegebenen Perioden am angegebenen Kanal. Zusätzlich wird das Puls-/Periodenverhältnis in [‰] angegeben.
PHASE (→ Seite 310)	liest ein Kanalpaar mit schnellen Eingängen ein und vergleicht die Phasenlage der Signale

ℹ Bei Einsatz dieser Bausteine werden automatisch die dort parametrierten Ein-/Ausgänge konfiguriert. Der Programmierer der Anwendung ist hiervon entlastet.

Hardware-Filter konfigurieren

9154

Über die Systemvariable Ixx_DFILTER kann ein digitaler Hardware-Filter an den schnellen Zähl- und Impulseingängen konfiguriert werden. Der Wert in µs (max. 100 000) gibt an, wie lange ein binärer Pegel ohne Unterbrechung anliegen muss, bevor er übernommen wird. Voreinstellung = 0 µs.

❗ Der Pegelwechsel des Eingangssignals wird um den im Filter eingestellten Wert verzögert.

Nur bei folgenden Funktionsbausteinen hat der Filter Auswirkungen auf die erfassten Signale:

FAST_COUNT (→ Seite 298)	Zählerbaustein für schnelle Eingangsimpulse
FREQUENCY (→ Seite 300)	misst die Frequenz des am gewählten Kanal ankommenden Signals
FREQUENCY_PERIOD (→ Seite 302)	misst die Frequenz und die Periodendauer (Zykluszeit) in [µs] am angegebenen Kanal
INC_ENCODER (→ Seite 304)	Vorwärts-/Rückwärts-Zählerfunktion zur Auswertung von Drehgebern
PERIOD (→ Seite 306)	misst am angegebenen Kanal die Frequenz und die Periodendauer (Zykluszeit) in [µs]
PERIOD_RATIO (→ Seite 308)	misst die Frequenz und die Periodendauer (Zykluszeit) in [µs] über die angegebenen Perioden am angegebenen Kanal. Zusätzlich wird das Puls-/Periodenverhältnis in [‰] angegeben.

Digitale Filter stehen nicht für alle schnellen Zähl- und Impulseingänge zur Verfügung.



Einsatz als Binäreingänge

3804

Durch die zulässigen hohen Eingangsfrequenzen können auch Fehlsignale erkannt werden, z.B. prellende Kontakte mechanischer Schalter.

► Bei Bedarf die Fehlsignale im Anwendungsprogramm unterdrücken!

Sichere Eingänge

12249

→ Kapitel ***Eingänge für Sicherheitsfunktionen*** (→ Seite 55)
(im Kapitel ***Hinweise für sicherheitsrelevante Anwendungen*** (→ Seite 15))

5.4.2 Ausgänge konfigurieren

Inhalt

3976

Zulässige Betriebsarten → Kapitel ***Mögliche Betriebsarten Ein-/Ausgänge*** (→ Seite 436)

Software-Filter der Ausgänge konfigurieren

6882

Über die Systemvariablen Qxx_FILTER kann ein Software-Filter konfiguriert werden, der die gemessenen Stromwerte filtert. Der Filter verhält sich wie ein klassischer Tiefpassfilter, wobei die Grenzfrequenz durch den in die Systemvariable eingetragenen Wert eingestellt wird.

Tabelle: Grenzfrequenz Software-Tiefpassfilter bei der Strommessung am Ausgang

Qxx_FILTER	Filterfrequenz [Hz]	Signalanstiegszeit	Hinweise
0	Filter deaktiviert		
1	580	0,6 ms	
2	220	1,6 ms	
3	102	3,5 ms	
4	51	7 ms	empfohlen, Voreinstellung
5	25	14 ms	
6	12	28 ms	
7	6	56 ms	
8	3	112 ms	
≥ 9	51	7 ms	→ Voreinstellung

Nach dem Ändern der Filtereinstellung wird der Wert dieses Ein- oder Ausgangs nicht sofort richtig ausgegeben. Erst nach der Signalanstiegszeit (→ Tabelle) ist der Wert wieder korrekt.

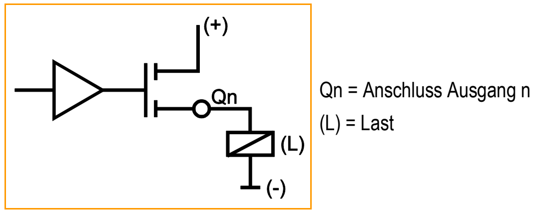
Die Signalanstiegszeit ist die Zeitdauer, die ein Signal am Ausgang des Filters benötigt, um von 10 % auf 90 % des Endwerts zu kommen, wenn am Eingang ein Sprung angelegt wird. Die Signalabstiegszeit ist die Zeitdauer von 90 % bis 10 %.

Binärausgänge: Konfiguration und Diagnose

13965

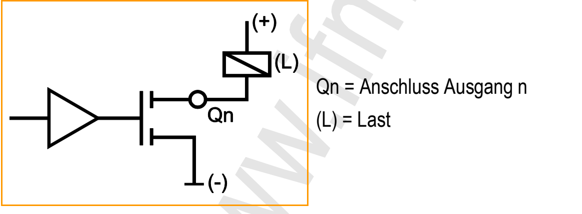
Bei den Geräte-Ausgängen sind folgende Betriebsarten möglich (→ Datenblatt):
- binärer Ausgang, plus-schaltend (BH) mit/ohne Diagnosefunktion
- binärer Ausgang plus-schaltend (BL) ohne Diagnosefunktion

15450



Qn = Anschluss Ausgang n
(L) = Last

Prinzipschaltung Binär-Ausgang plus-schaltend (BH) für positives Ausgangssignal



Qn = Anschluss Ausgang n
(L) = Last

Prinzipschaltung Binär-Ausgang minus-schaltend (BL) für negatives Ausgangssignal

► Betriebsarten mit folgendem Funktionsbaustein einstellen:

SET_OUTPUT_MODE (→ Seite 313)	setzt die Betriebsart des gewählten Ausgangskanals

⚠ WARNUNG

Gefährlicher Wiederanlauf möglich!
Gefahr von Personenschaden! Gefahr von Sachschaden an der Maschine/Anlage!

Wird ein Ausgang im Fehlerfall hardwaremäßig abgeschaltet, ändert sich der durch das Anwendungsprogramm erzeugte logische Zustand dadurch nicht.

► Abhilfe:
- Die Ausgänge zunächst im Anwendungsprogramm logisch zurücksetzen!
- Fehler beseitigen!
- Ausgänge situationsabhängig wieder setzen.

HINWEIS

► Im laufenden Betrieb die Ausgänge NICHT umkonfigurieren!
Ändern von PWM-Ausgang nach Binär-Ausgang ist nicht zulässig.

> Ansonsten könnten die Ausgänge unvorhersehbar reagieren.

► Ausgang als Binärausgang mit Diagnose nutzen:
→ FB SET_OUTPUT_MODE > Eingang DIAGNOSTICS = TRUE

> Leiterbruch und Kurzschluss des Ausgangssignals werden (gebündelt je Ausgangsgruppe) getrennt über die Systemvariablen ERROR_BREAK_Qx oder ERROR_SHORT_Qx angezeigt (→ Kapitel ***Systemmerker*** (→ Seite 420)).

► Die einzelnen Ausgangs-Fehlerbits bei Bedarf im Anwendungsprogramm ausmaskieren.

Ausgangssignale filtern

13521

→ Kapitel ***Software-Filter der Ausgänge konfigurieren*** (→ Seite 184)

Sichere binäre Ausgänge

13966

→ Kapitel ***Ausgänge für Sicherheitsfunktionen*** (→ Seite 65) (dort auch: Diagnose der Ausgänge)
(im Kapitel ***Hinweise für sicherheitsrelevante Anwendungen*** (→ Seite 15))

⚠ WARNUNG

Sach- oder Körperschäden möglich durch Fehlfunktionen!

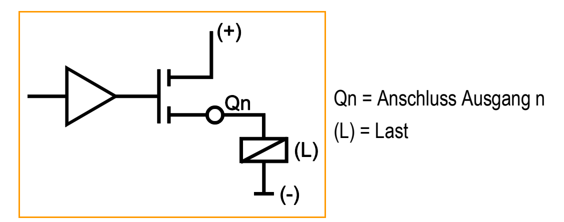
► Für sicherheitsrelevante Anwendungen die Ausgänge verwenden, die als sichere Ausgänge konfiguriert werden können!

PWM-Ausgänge

13968

Bei den Geräte-Ausgängen sind folgende Betriebsarten möglich (→ Datenblatt):
- PWM-Ausgang, plus-schaltend (BH) ohne Diagnosefunktion
- PWM-Ausgangspaar H-Brücke ohne Diagnosefunktion

15451



Qn = Anschluss Ausgang n
(L) = Last

Prinzipschaltung Binär-Ausgang plus-schaltend (BH) für positives Ausgangssignal

14713

⚠ WARNUNG

Sach- oder Körperschäden möglich durch Fehlfunktionen!

Für Ausgänge im PWM-Modus gilt:
- es gibt keine Diagnosefunktionen
- es werden keine ERROR-Merker gesetzt
- der Überlastschutz OUT_OVERLOAD_PROTECTION ist NICHT aktiv

HINWEIS

► Im laufenden Betrieb die Ausgänge NICHT umkonfigurieren!
Ändern von PWM-Ausgang nach Binär-Ausgang ist nicht zulässig.

> Ansonsten könnten die Ausgänge unvorhersehbar reagieren.

- PWM-Ausgänge können mit und ohne Stromregelfunktion betrieben werden.
  Stromgeregelte PWM-Ausgänge werden überwiegend zur Ansteuerung von proportionalen Hydraulikfunktionen genutzt.
- In diesen Geräten können bis zu 2 H-Brücken (ExtendedController: bis zu 4 H-Brücken) z.B. zur Ansteuerung von Elektromotoren realisiert werden.

Verfügbarkeit von PWM

13971

Gerät	Anzahl verfügbare PWM-Ausgänge	davon stromgeregelt (PWMi)	PWM-Frequenz [Hz]
SafetyController: CR7032	16	16	20...250
ExtendedSafetyController: CR7132	32	32	20...250

FBs für PWM-Funktionen

14710

Für die PWM-Funktion der Ausgänge stehen folgende Funktionsbausteine zur Verfügung:

OUTPUT_BRIDGE (→ Seite 326)	H-Brücke an einem PWM-Kanalpaar
OUTPUT_CURRENT (→ Seite 330)	misst den Strom (Mittelung über Dither-Periode) an einem Ausgangskanal
OUTPUT_CURRENT_CONTROL (→ Seite 331)	Stromregler für einen PWMi-Ausgangskanal
PWM1000 (→ Seite 333)	initialisiert und parametriert einen PWM-fähigen Ausgangskanal das Puls-Pausen-Verhältnis kann in 1 ‰-Schritten angegeben werden

Mögliche Spezial-Funktionen der Ausgänge:
→ Kapitel ***Bausteine: Hydraulikregelung*** (→ Seite 335)
→ Kapitel ***Bausteine: Regler*** (→ Seite 351)

Stromregelung mit PWM (= PWMi)

13829

Über die im Controller integrierten Strommesskanäle kann eine Strommessung des Spulenstroms durchgeführt werden. Dadurch kann zum Beispiel der Strom bei einer Spulenerwärmung nachgeregelt werden. Damit bleiben die Hydraulikverhältnisse im System gleich.

Grundsätzlich sind die stromgeregelten Ausgänge gegen Kurzschluss geschützt.

Ausgangssignale filtern

13521

→ Kapitel ***Software-Filter der Ausgänge konfigurieren*** (→ Seite 184)

Sichere PWM-Ausgänge

13834

Dieses Gerät bietet keine sicherheitsfähigen PWM-Ausgänge.

5.5 Variablen

Inhalt

3130

In diesem Kapitel erfahren Sie mehr über den Umgang mit Variablen.
Grundsätzlich unterscheiden wir in CODESYS folgende Arten von Variablen:

Variable	Deklaration	Gültigkeitsbereich	Speicherverhalten

lokal	im Baustein:
	`PROGRAM...`
	`VAR`
	`(Deklaration)`
	`END_VAR`
		Gilt nur im Baustein (POU), wo sie konfiguriert wurde.	flüchtig

global	in [Ressourcen] > [Globale Variablen] > [Globale_Variablen]:
	`VAR_GLOBAL`
	`(Deklaration)`
	`END_VAR`
		Gilt in allen Bausteinen (POUs) dieses Projekts.	

lokal Retain	im Bausteins:
	`PROGRAM...`
	`VAR RETAIN`
	`(Deklaration)`
	`END_VAR`
		Gilt nur im Baustein (POU), wo sie konfiguriert wurde.	Werte werden im FRAM-Speicher gesichert ¹)

global Retain	in [Ressourcen] > [Globale Variablen] > [Globale_Variablen]:
	`VAR_GLOBAL RETAIN`
	`(Deklaration)`
	`END_VAR`
		Gilt in allen Bausteinen (POUs) dieses CODESYS-Projekts.	

Netzwerk	in [Ressourcen] > [Globale Variablen] > Deklarationsliste:
	`VAR_GLOBAL`
	`(Deklaration)`
	`END_VAR`
		Werte stehen allen CODESYS-Projekten im gesamten Netzwerk zur Verfügung, wenn die Variable in ihren Deklarationslisten enthalten ist.	flüchtig

Netzwerk Retain	in [Ressourcen] > [Globale Variablen] > Deklarationsliste:
	`VAR_GLOBAL RETAIN`
	`(Deklaration)`
	`END_VAR`
			Werte werden im FRAM-Speicher gesichert ¹)

¹) FRAM steht hier allgemein für alle Arten von nichtflüchtigen, schnellen Speichern.

Im Folgenden beschreiben wir die Besonderheiten für ...

- ***Retain-Variablen*** (→ Seite 189) und
- ***Netzwerkvariablen*** (→ Seite 189).

→ CODESYS-Programmierhandbuch → ecomatmobile-DVD "Software, tools and documentation"

5.5.1 Retain-Variablen

15454

Als RETAIN deklarierte Variablen erzeugen remanente Daten. Retain-Variablen behalten beim Aus-/Einschalten des Geräts oder einem Online-Reset die in ihnen gespeicherten Werte.

> ❗ Die Inhalte der Retain-Variablen gehen verloren, falls beim Ausschalten das Gerät im STOP-Zustand ist!

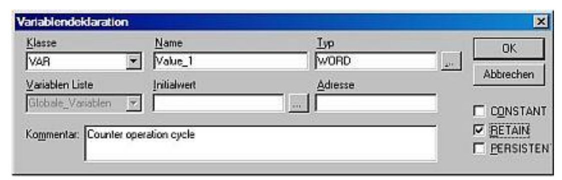
14166

Typische Einsätze für Retain-Variablen sind z.B.:
- Betriebsstunden, die zur Laufzeit der Maschine fortgeschrieben werden,
- Positionswerte von Inkrementalgebern,
- im Bildschirmgerät eingetragene Sollwerte,
- Maschinenparameter,

also alle Variablen, deren Werte beim Ausschalten des Geräts nicht verloren gehen dürfen.

Als Retain können alle Variablentypen, auch komplexe Stukturen (z.B. Timer), gekennzeichnet werden.

► Dazu in der Variablen-Deklaration das Kontrollfeld [RETAIN] aktivieren (→ Bild).



5.5.2 Netzwerkvariablen

9856

Globale Netzwerkvariablen dienen dem Datenaustausch zwischen Controllern im Netzwerk. Die Werte von globalen Netzwerkvariablen stehen allen CODESYS-Projekten im gesamten Netzwerk zur Verfügung, wenn die Variablen in deren Deklarationslisten enthalten sind.

► Dazu folgende Bibliothek(en) in das CODESYS-Projekt einbinden:
  - 3S_CANopenNetVar.lib

5.5.3 Für sicherheitsrelevante Daten zulässige Variablen

14271

→ Kapitel ***Variablen: zulässig für sicherheitsrelevante Daten*** (→ Seite 111)

6 ifm-Funktionselemente

Inhalt

13586

Alle CODESYS-Funktionselemente (FBs, PRGs, FUNs) sind in Bibliotheken zusammengefasst. Nachfolgend zeigen wir Ihnen alle **ifm**-Bibliotheken, die Sie zusammen mit diesem Gerät nutzen können.

Anschließend finden Sie eine thematisch gegliederte Beschreibung der Funktionselemente.

6.1 ifm-Bibliotheken für das Gerät CR7132

Inhalt

13585

Legende für ..._Vxxyyzz.LIB:

V	Version
xx: 00...99	Versionsnummer
yy: 00...99	Release-Nummer
zz: 00...99	Patch-Nummer

Hier finden Sie die für dieses Gerät passenden **ifm**-Funktionselemente aufgelistet, nach CODESYS-Bibliotheken sortiert.

Für sicherheitsrelevante Funktionen zulässige Bibliotheken und Bausteine:
→ Kapitel ***Funktionsbausteine: zulässig für sicherheitsrelevante Funktionen*** (→ Seite 100)

6.1.1 Bibliothek ifm_CR7132_Vxxyyzz.LIB

14238

Dies ist die Geräte-Bibliothek.
Diese **ifm**-Bibliothek enthält folgende Bausteine:

Baustein	Kurzbeschreibung
CANx (→ Seite 203)	initialisiert die CAN-Schnittstelle x x = 1...n = Nummer der CAN-Schnittstelle (je nach Gerät, → Datenblatt)
CANx_BAUDRATE (→ Seite 204)	stellt die Übertragungsrate für den Busteilnehmer an der CAN-Schnittstelle x ein x = 1...n = Nummer der CAN-Schnittstelle (je nach Gerät, → Datenblatt)
CANx_BUSLOAD (→ Seite 205)	ermittelt die aktuelle Buslast an der CAN-Schnittstelle x und zählt die aufgetretenen Error-Frames x = 1...n = Nummer der CAN-Schnittstelle (je nach Gerät, → Datenblatt)
CANx_DOWNLOADID (→ Seite 207)	stellt den Download-Identifier für die CAN-Schnittstelle x ein x = 1...n = Nummer der CAN-Schnittstelle (je nach Gerät, → Datenblatt)
CANx_ERRORHANDLER (→ Seite 208)	führt ein "manuelles" Bus-Recover auf der CAN-Schnittstelle x durch x = 1...n = Nummer der CAN-Schnittstelle (je nach Gerät, → Datenblatt)
CANx_RECEIVE (→ Seite 209)	CAN-Schnittstelle x: konfiguriert ein Datenempfangsobjekt und liest den Empfangspuffer des Datenobjektes aus x = 1...n = Nummer der CAN-Schnittstelle (je nach Gerät, → Datenblatt)
CANx_SDO_READ (→ Seite 239)	CAN-Schnittstelle x: liest das SDO mit den angegebenen Indizes aus dem Knoten aus x = 1...n = Nummer der CAN-Schnittstelle (je nach Gerät, → Datenblatt)
CANx_SDO_WRITE (→ Seite 241)	CAN-Schnittstelle x: schreibt das SDO mit den angegebenen Indizes in den Knoten x = 1...n = Nummer der CAN-Schnittstelle (je nach Gerät, → Datenblatt)
CANx_TRANSMIT (→ Seite 211)	übergibt in jedem Aufruf ein CAN-Datenobjekt (Message) an die CAN-Schnittstelle x zur Übertragung x = 1...n = Nummer der CAN-Schnittstelle (je nach Gerät, → Datenblatt)
CAN_SAFETY_RECEIVE (→ Seite 214)	empfängt eine sichere CAN-Nachricht (SRDO)
CAN_SAFETY_TRANSMIT (→ Seite 217)	überträgt eine sichere CAN-Nachricht (SRDO)
CHECK_DATA (→ Seite 375)	erzeugt über einen konfigurierbaren Speicherbereich eine Prüfsumme (CRC) und prüft die Daten des Speicherbereichs auf ungewollte Veränderung
DELAY (→ Seite 353)	verzögert die Ausgabe des Eingangswertes um die Zeit T (Totzeit-Glied)
ERROR_REPORT (→ Seite 382)	meldet dem System einen anwendungsspezifischen Fehler
ERROR_RESET (→ Seite 384)	setzt anstehende Fehlermeldungen zurück
FAST_COUNT (→ Seite 298)	Zählerbaustein für schnelle Eingangsimpulse
FAST_COUNT_E	= FAST_COUNT für die Extended-Seite
FLASHREAD (→ Seite 367)	liest unterschiedliche Datentypen direkt aus dem Flash-Speicher in den RAM
FLASHWRITE (→ Seite 368)	schreibt unterschiedliche Datentypen direkt in den Flash-Speicher
FRAMREAD (→ Seite 370)	liest unterschiedliche Datentypen direkt aus dem FRAM-Speicher in den RAM FRAM steht hier allgemein für alle Arten von nichtflüchtigen, schnellen Speichern.
FRAMWRITE (→ Seite 371)	schreibt unterschiedliche Datentypen direkt in den FRAM-Speicher FRAM steht hier allgemein für alle Arten von nichtflüchtigen, schnellen Speichern.
FREQUENCY (→ Seite 300)	misst die Frequenz des am gewählten Kanal ankommenden Signals
FREQUENCY_E	= FREQUENCY für die Extended-Seite
FREQUENCY_PERIOD (→ Seite 302)	misst die Frequenz und die Periodendauer (Zykluszeit) in [µs] am angegebenen Kanal
FREQUENCY_PERIOD_E	= FREQUENCY_PERIOD für die Extended-Seite

Baustein	Kurzbeschreibung
GET_IDENTITY (→ Seite 377)	liest die im Gerät gespeicherten spezifischen Kennungen: • Hardware-Name und Hardware-Version des Geräts • Seriennummer des Geräts • Name des Laufzeitsystems im Gerät • Version und Ausgabe des Laufzeitsystems im Gerät • Name der Anwendung (wurde zuvor mit ***SET_IDENTITY*** (→ Seite 379) gespeichert)
INC_ENCODER (→ Seite 304)	Vorwärts-/Rückwärts-Zählerfunktion zur Auswertung von Drehgebern
INC_ENCODER_E	= INC_ENCODER für die Extended-Seite
INPUT_ANALOG (→ Seite 261)	Strom- und Spannungsmessung am analogen Eingangskanal
INPUT_ANALOG_E	= INPUT_ANALOG für die Extended-Seite
MEMCPY (→ Seite 372)	schreibt und liest unterschiedliche Datentypen direkt in den Speicher
MEMORY_RETAIN_PARAM (→ Seite 365)	legt das remanente Verhalten der Daten für verschiedene Ereignisse fest
MEMSET (→ Seite 373)	beschreibt einen bestimmten Datenbereich
NORM (→ Seite 293)	normiert einen Wert [WORD] innerhalb festgelegter Grenzen auf einen Wert mit neuen Grenzen
NORM_DINT (→ Seite 295)	normiert einen Wert [DINT] innerhalb festgelegter Grenzen auf einen Wert mit neuen Grenzen
NORM_REAL (→ Seite 296)	normiert einen Wert [REAL] innerhalb festgelegter Grenzen auf einen Wert mit neuen Grenzen
OUTPUT_BRIDGE (→ Seite 326)	H-Brücke an einem PWM-Kanalpaar
OUTPUT_BRIDGE_E	= OUTPUT_BRIDGE für die Extended-Seite
OUTPUT_CURRENT (→ Seite 330)	misst den Strom (Mittelung über Dither-Periode) an einem Ausgangskanal
OUTPUT_CURRENT_E	= OUTPUT_CURRENT für die Extended-Seite
OUTPUT_CURRENT_CONTROL (→ Seite 331)	Stromregler für einen PWMi-Ausgangskanal
OUTPUT_CURRENT_CONTROL_E	= OUTPUT_CURRENT_CONTROL für die Extended-Seite
PACK_ERRORCODE (→ Seite 388)	hilft beim Zusammenbauen eines ERRORCODE aus den Bytes für: • Fehlerklasse • anwendungsspezifischer Fehler • Fehlerquelle • Fehlerursache
PERIOD (→ Seite 306)	misst am angegebenen Kanal die Frequenz und die Periodendauer (Zykluszeit) in [µs]
PERIOD_E	= PERIOD für die Extended-Seite
PERIOD_RATIO (→ Seite 308)	misst die Frequenz und die Periodendauer (Zykluszeit) in [µs] über die angegebenen Perioden am angegebenen Kanal. Zusätzlich wird das Puls-/Periodenverhältnis in [‰] angegeben.
PERIOD_RATIO_E	= PERIOD_RATIO für die Extended-Seite
PHASE (→ Seite 310)	liest ein Kanalpaar mit schnellen Eingängen ein und vergleicht die Phasenlage der Signale
PHASE_E	= PHASE für die Extended-Seite
PID2 (→ Seite 354)	PID-Regler
PT1 (→ Seite 356)	Regelstrecke mit Verzögerung 1. Ordnung
PWM1000 (→ Seite 333)	initialisiert und parametriert einen PWM-fähigen Ausgangskanal das Puls-Pausen-Verhältnis kann in 1 ‰-Schritten angegeben werden
PWM1000_E	= PWM1000 für die Extended-Seite
SAFETY_SWITCH (→ Seite 268)	Betrieb der 1-kanaligen SafetySwitch der ifm electronic gmbh
SERIAL_PENDING (→ Seite 256)	ermittelt die Anzahl der im seriellen Empfangspuffer gespeicherten Datenbytes
SERIAL_RX (→ Seite 257)	liest mit jedem Aufruf ein empfangenes Datenbyte aus dem seriellen Empfangspuffer aus
SERIAL_SETUP (→ Seite 258)	initialisiert die serielle RS232-Schnittstelle
SERIAL_TX (→ Seite 259)	überträgt ein Datenbyte über die serielle RS232-Schnittstelle
SET_DEBUG (→ Seite 378)	organisiert (abhängig vom TEST-Eingang) den DEBUG-Modus oder den Monitoring-Modus
SET_IDENTITY (→ Seite 379)	setzt eine anwendungsspezifische Programmkennung

Baustein	Kurzbeschreibung
SET_INPUT_MODE (→ Seite 264)	weist einem Eingangskanal eine Betriebsart zu
SET_INPUT_MODE_E	= SET_INPUT_MODE für die Extended-Seite
SET_KEEP_ALIVE (→ Seite 386)	konfiguriert, welcher Ausgangskanal und welcher CANsafety-Kanal beim Auftreten eines bestimmten schweren Fehlers weiterbetrieben werden sollen, da sie von dem mit ERRORCODE gemeldeten Fehler unabhängig sind
SET_OUTPUT_MODE (→ Seite 313)	setzt die Betriebsart des gewählten Ausgangskanals
SET_OUTPUT_MODE_E	= SET_OUTPUT_MODE für die Extended-Seite
SET_PASSWORD (→ Seite 380)	setzt Benutzerkennung für Zugangskontrolle bei Programm- und Speicher-Upload
SHOW_ERROR_LIST (→ Seite 389)	liest den aktuell vorliegenden Fehler-Code
TEMPERATURE (→ Seite 361)	liest die aktuelle Temperatur im Gerät aus
TIMER_READ (→ Seite 358)	liest die aktuelle Systemzeit in [ms] aus Max-Wert = 49d 17h 2min 47s 295ms
TIMER_READ_US (→ Seite 359)	liest die aktuelle Systemzeit in [µs] aus Max-Wert = 1h 11min 34s 967ms 295µs
UNPACK_ERRORCODE (→ Seite 390)	hilft beim Trennen eines ERRORCODE in die Bytes für: • Fehlerklasse • anwendungsspezifischer Fehler • Fehlerquelle • Fehlerursache

6.1.2 Bibliothek ifm_CR7132_CANopenxMaster_Vxxyyzz.LIB

13707

x = 1...4 = Nummer der CAN-Schnittstelle
Diese Bibliothek enthält Bausteine für den Betrieb des Geräts als CANopen-Master.

Diese **ifm**-Bibliothek enthält folgende Bausteine:

Baustein	Kurzbeschreibung
CANx_MASTER_EMCY_HANDLER (→ Seite 221)	verwaltet den geräteeigenen Fehlerstatus des CANopen-Masters an der CAN-Schnittstelle x x = 1...n = Nummer der CAN-Schnittstelle (je nach Gerät, → Datenblatt)
CANx_MASTER_SEND_EMERGENCY (→ Seite 222)	versendet anwendungsspezifische Fehlerstatus des CANopen-Masters an der CAN-Schnittstelle x x = 1...n = Nummer der CAN-Schnittstelle (je nach Gerät, → Datenblatt)
CANx_MASTER_STATUS (→ Seite 224)	Status-Anzeige an der CAN-Schnittstelle x des als CANopen-Master eingesetzten Gerätes x = 1...n = Nummer der CAN-Schnittstelle (je nach Gerät, → Datenblatt)

6.1.3 Bibliothek ifm_CR7132_CANopenxSlave_Vxxyyzz.LIB

13709

x = 1...4 = Nummer der CAN-Schnittstelle
Diese Bibliothek enthält Bausteine für den Betrieb des Geräts als CANopen-Slave.

Diese **ifm**-Bibliothek enthält folgende Bausteine:

Baustein	Kurzbeschreibung
CANx_SLAVE_EMCY_HANDLER (→ Seite 231)	verwaltet den geräteeigenen Fehlerstatus des CANopen-Slaves an der CAN-Schnittstelle x: • Error Register (Index 0x1001) und • Error Field (Index 0x1003) des CANopen Objektverzeichnis x = 1...n = Nummer der CAN-Schnittstelle (je nach Gerät, → Datenblatt)
CANx_SLAVE_NODEID (→ Seite 232)	ermöglicht das Einstellen der Node-ID eines CANopen-Slaves an der CAN-Schnittstelle x zur Laufzeit des Anwendungsprogramms x = 1...n = Nummer der CAN-Schnittstelle (je nach Gerät, → Datenblatt)
CANx_SLAVE_SEND_EMERGENCY (→ Seite 233)	versendet anwendungsspezifische Fehlerstatus des CANopen-Slaves an der CAN-Schnittstelle x x = 1...n = Nummer der CAN-Schnittstelle (je nach Gerät, → Datenblatt)
CANx_SLAVE_SET_PREOP (→ Seite 235)	schaltet den Betriebsmodus dieses CANopen-Slaves an der CAN-Schnittstelle x von OPERATIONAL auf PRE-OPERATIONAL x = 1...n = Nummer der CAN-Schnittstelle (je nach Gerät, → Datenblatt)
CANx_SLAVE_STATUS (→ Seite 236)	zeigt den Status des an der CAN-Schnittstelle x als CANopen-Slave eingesetzten Gerätes x = 1...n = Nummer der CAN-Schnittstelle (je nach Gerät, → Datenblatt)

6.1.4 Bibliothek ifm_CR7132_J1939_Vxxyyzz.LIB

13711

Diese Bibliothek enthält Bausteine zur Motorsteuerung.

Diese **ifm**-Bibliothek enthält folgende Bausteine:

Baustein	Kurzbeschreibung
J1939_x (→ Seite 244)	CAN-Schnittstelle x: Protokoll-Handler für das Kommunikationsprofil SAE J1939 x = 1...n = Nummer der CAN-Schnittstelle (je nach Gerät, → Datenblatt)
J1939_x_GLOBAL_REQUEST (→ Seite 245)	CAN-Schnittstelle x: organisiert globales Anfordern und Empfangen von Daten der J1939-Netzwerkteilnehmer x = 1...n = Nummer der CAN-Schnittstelle (je nach Gerät, → Datenblatt)
J1939_x_RECEIVE (→ Seite 247)	CAN-Schnittstelle x: empfängt eine einzelne Nachricht oder einen Nachrichtenblock x = 1...n = Nummer der CAN-Schnittstelle (je nach Gerät, → Datenblatt)
J1939_x_RESPONSE (→ Seite 249)	CAN-Schnittstelle x: organisiert die automatische Antwort auf ein Request-Telegramm x = 1...n = Nummer der CAN-Schnittstelle (je nach Gerät, → Datenblatt)
J1939_x_SPECIFIC_REQUEST (→ Seite 251)	CAN-Schnittstelle x: automatisches Anfordern einzelner Nachrichten von einem bestimmten (specific) J1939-Netzwerkteilnehmer x = 1...n = Nummer der CAN-Schnittstelle (je nach Gerät, → Datenblatt)
J1939_x_TRANSMIT (→ Seite 253)	CAN-Schnittstelle x: versendet einzelne Nachrichten oder Nachrichtenblocks x = 1...n = Nummer der CAN-Schnittstelle (je nach Gerät, → Datenblatt)

6.1.5 Bibliothek ifm_hydraulic_32bit_Vxxyyzz.LIB

13729

Diese Bibliothek enthält Bausteine für Hydraulik-Steuerungen.

Diese **ifm**-Bibliothek enthält folgende Bausteine:

Baustein	Kurzbeschreibung
CONTROL_OCC (→ Seite 336)	OCC = **O**utput **C**urrent **C**ontrol (= stromgeregelter Ausgang) skaliert den Eingangswert [WORD] auf einen angegebenen Strombereich
JOYSTICK_0 (→ Seite 339)	skaliert Signale [INT] aus einem Joystick auf fest definierte Kennlinien, normiert auf 0…1000
JOYSTICK_1 (→ Seite 342)	skaliert Signale [INT] aus einem Joystick auf parametrierbare Kennlinien, normiert auf 0…1000
JOYSTICK_2 (→ Seite 346)	skaliert Signale [INT] aus einem Joystick auf einen parametrierbaren Kennlinien-Verlauf; die Normierung ist frei bestimmbar
NORM_HYDRAULIC (→ Seite 349)	normiert einen Wert [DINT] innerhalb festgelegter Grenzen auf einen Wert mit neuen Grenzen

6.1.6 Bibliothek ifm_SafetyPLCopen_Vxxyyzz.LIB

13727

Diese Bibliothek enthält zertifizierte Bausteine für sicherheitsrelevante Anwendungen.

Diese ifm-Bibliothek enthält folgende Bausteine:

Baustein	Kurzbeschreibung
SF_ANTIVALENT (→ Seite 272)	• vergleicht zwei binäre sichere Eingänge miteinander • prüft den zeitlichen Ablauf der Eingänge zueinander
SF_EMERGENCYSTOP (→ Seite 274)	überwacht einen Not-Halt-Schalter
SF_ENABLESWITCH (→ Seite 277)	dient zum Auswerten der Signale einer Freigabetaste mit drei Schaltstufen
SF_ENABLESWITCH_2 (→ Seite 280)	dient zum Auswerten der Signale einer Freigabetaste mit zwei oder drei Schaltstufen
SF_EQUIVALENT (→ Seite 283)	• vergleicht zwei binäre sichere Eingänge miteinander • prüft den zeitlichen Ablauf der Eingänge zueinander
SF_EQUIVALENT_REAL (→ Seite 285)	• vergleicht zwei sichere Eingangswerte [REAL] miteinander • prüft die Werte auf zulässigen Wertebereich und zulässige Abweichung
SF_EQUIVALENT_WORD (→ Seite 287)	• vergleicht zwei sichere Eingangswerte [WORD] miteinander • prüft die Werte auf zulässigen Wertebereich und zulässige Abweichung
SF_MODESELECTOR (→ Seite 199)	ermöglicht das sichere Schalten zwischen bis zu 8 Betriebsarten einer Maschine oder Anlage
SF_OUTCONTROL (→ Seite 319)	kontrolliert einen sicheren Ausgang mit einem Signal aus der funktionellen Anwendung und einem sicheren Signal mit optionaler Anlaufsperre
SF_SAFETYREQUEST (→ Seite 322)	stellt eine Schnittstelle zu einem allgemeinen Aktuator zur Verfügung, um den Aktuator in den sicheren Zustand zu setzen
SF_TWOHANDCONTROL (→ Seite 289)	realisiert eine Zweihandbedienung

6.2 ifm-Bausteine für das Gerät CR7132

Inhalt

14255
3826

Hier finden Sie die Beschreibung der für dieses Gerät passenden **ifm**-Funktionselemente, nach Thema sortiert.

Für sicherheitsrelevante Funktionen zulässige Bibliotheken und Bausteine:
→ Kapitel ***Funktionsbausteine: zulässig für sicherheitsrelevante Funktionen*** (→ Seite 100)

6.2.1 Bausteine: Betriebsarten sicher umschalten

Inhalt

13752

Für den sicheren Betrieb einer Maschine oder Anlage unterstützen folgende Funktionen das sichere Umschalten der System-Betriebsarten.

Legende zu den Ein- und Ausgängen der Sicherheits-FBs 'SF_...'

12700

- Alle Ein- und Ausgänge der Sicherheits-FBs 'SF_...' werden beim Hochlaufen (Booten) der Steuerung mit Initialwerten belegt. So werden undefinierte Zustände der FBs verhindert. Die Initialwerte führen immer zum sicheren Zustand der Steuerung.
- Alle Sicherheits-FBs arbeiten nur, wenn der Eingang ENABLE=TRUE ist.
- Der Ausgang READY=TRUE gibt an, dass der FB aktiv ist.
- Der Ausgang SAFETYDEMAND=TRUE signalisiert, dass Handlungsbedarf durch den Bediener erforderlich ist.
- Der Eingang S_SAFETYACTIVE=TRUE signalisiert dem FB, dass der relevante Prozess im Sicherheitsmodus ist.
- Der Ausgang RESETREQUEST=TRUE fordert vom Bediener:
  • gegebenenfalls die falschen Eingangssignale beheben
  • mit RESET-Signal bestätigen, dass der FB den sicheren Prozess fortsetzen darf
  Andernfalls verbleibt der FB im Warten- oder Fehler-Zustand.
  FBs ohne Eingang RESET: der Ausgang RESETREQUEST ist ohne Funktion.
- Der Eingang RESET=TRUE (nur 1 Zyklus) signalisiert dem FB:
  • den sicheren Prozess fortsetzen, sofern die Fehler beseitigt sind.
  Dauerhafter RESET führt zum Fehler.
- Der Ausgang ERROR=TRUE signalisiert einen Fehler.
  Der Fehler-Code erscheint im Ausgang DIAGCODE mit Werten ab 0xC000.
- Der Ausgang DIAGCODE liefert permanent Informationen über den Status des FBs.

SF_MODESELECTOR

12497

Baustein-Typ = Funktionsbaustein (FB)

Baustein ist enthalten in Bibliothek `ifm_SafetyPLCopen_Vxxyyzz.lib`

Symbol in CODESYS:

SF_MODESELECTOR	
ENABLE	READY
S_MODE0	S_MODE0SEL
S_MODE1	S_MODE1SEL
S_MODE2	S_MODE2SEL
S_MODE3	S_MODE3SEL
S_MODE4	S_MODE4SEL
S_MODE5	S_MODE5SEL
S_MODE6	S_MODE6SEL
S_MODE7	S_MODE7SEL
S_UNLOCK	S_ANYMODESEL
S_SETMODE	SAFETYDEMAND
AUTOSETMODE	RESETREQUEST
MODEMONITORTIME	ERROR
RESET	DIAGCODE

Beschreibung

12500

Der FB SF_MODESELECTOR ermöglicht das sichere Schalten zwischen bis zu 8 Betriebsarten einer Maschine oder Anlage.

Ablauf:

1. aktuelle Betriebsart entsperren
2. neue Betriebsart wählen
3. neu gewählte Betriebsart verriegeln (abwählbar)

Der FB überwacht den Ablauf und die Dauer des Umschaltvorgangs.

Parameter der Eingänge

12502

Parameter	Datentyp	Beschreibung
ENABLE	BOOL	TRUE: Baustein ausführen FALSE: Baustein wird nicht ausgeführt > Baustein-Eingänge sind nicht aktiv > Baustein-Ausgänge sind nicht spezifiziert
S_MODEx	BOOL	TRUE: Betriebsart x wurde angefordert (x=0...7) FALSE (= Initialwert): Betriebsart x wurde nicht angefordert
S_UNLOCK	BOOL	Entriegelt die aktuelle Betriebsart TRUE: Der aktuell aktive Ausgang S_MODExSEL (x=0...7) wird entriegelt, Änderungen sind möglich FALSE (= Initialwert): Der aktuell aktive Ausgang S_MODExSEL wird verriegelt, Änderungen an irgend einem Eingang S_MODEx führen zu keiner Änderung an den Ausgängen, auch nicht bei einer steigenden Flanke am Eingang S_SETMODE.
S_SETMODE	BOOL	Betriebsartänderung bestätigen FALSE ⇨ TRUE (Flanke): > Der zum gewählten Betriebsmode S_MODEx passende Ausgang S_MODExSEL wird TRUE (x=0...7) FALSE: im weiteren Programmablauf (= Initialwert)
AUTOSETMODE	BOOL	Konfiguration der Umschalt-Bestätigung TRUE: Ein gültiger Wechsel von einem S_MODEx Eingang zu einem anderen führt direkt zu einen Änderung an den S_MODExSEL Ausgängen (vorausgesetzt S_UNLOCK=TRUE) FALSE (= Initialwert): Eine Betriebsartenänderung muss mit dem Eingang S_SET_MODE bestätigt werden. Erst dann wird der zugehörige Ausgang S_MODExSEL=TRUE
MODEMONITORTIME	TIME	maximal zulässige Zeit für einen Wechsel an den Eingängen S_MODEx (x=0...7), während der alle Eingänge S_MODEx den Wert FALSE haben dürfen Initialwert = T#0ms
RESET	BOOL	FALSE ⇨ TRUE: Baustein zurücksetzen nach einem Fehler sonst: diese Funktion wird nicht ausgeführt (= Initialwert)

Parameter der Ausgänge

12508

Parameter	Datentyp	Beschreibung
READY	BOOL	TRUE: Funktionsbaustein ist aktiv Die Werte an den FB-Ausgängen sind gültig FALSE (= Initialwert): Funktionsbaustein ist nicht aktiv
S_MODExSEL	BOOL	TRUE: Betriebsart x gewählt und bestätigt (x=0...7) FALSE (= Initialwert): Betriebsart x nicht gewählt oder nicht bestätigt
S_ANYMODESEL	BOOL	TRUE: eine der 8 Betriebsarten ist gewählt und bestätigt FALSE (= Initialwert): keine Betriebsart ist gewählt oder bestätigt Sobald ein anderer S_MODEx=TRUE als der zuletzt gewählte: > S_ANYMODESEL=FALSE > der bisher gewählte und aktivierte S_MODExSEL=FALSE
SAFETYDEMAND	BOOL	TRUE: Sicherheitsfunktion ist angefordert Aktion des Maschinenführers erforderlich FALSE (= Initialwert): Sicherheitsfunktion nicht angefordert
RESETREQUEST	BOOL	Ein Reset ist erforderlich, damit der FB weiterarbeiten kann TRUE: Reset ist erforderlich FALSE (= Initialwert): kein Reset erforderlich
ERROR	BOOL	TRUE: ein Fehler ist aufgetreten FB ist im Fehlerzustand FALSE (= Initialwert): kein Fehler aufgetreten
DIAGCODE	WORD	Informationen zum aktuellen Zustand des FBs (mögliche Meldungen → folgende Tabelle)

Mögliche Ergebnisse für DIAGCODE:

Wert [hex]	Beschreibung
0000	(Initialwert) Funktionsbaustein ist nicht aktiv
8000	Sicherheitsausgang ist freigegeben
8004	Betriebsart ist gültig und verriegelt
8005	Betriebsart ist gültig aber nicht verriegelt
C001	Parametrierfehler: mehrere Auswahl-Eingänge = TRUE FB wartet auf RESET
C002	Überwachungszeit abgelaufen (FB wartet auf Auswahl-Eingang) FB wartet auf RESET
C003	im Zustand C001 ist RESET-Signal statisch FB wartet auf RESET=FALSE
C004	im Zustand C002 ist RESET-Signal statisch FB wartet auf RESET=FALSE

6.2.2 Bausteine: CAN Layer 2

Inhalt

13754

Hier werden die CAN-Funktionsbausteine (Layer 2) zur Nutzung im Anwendungsprogramm beschrieben.

CANx

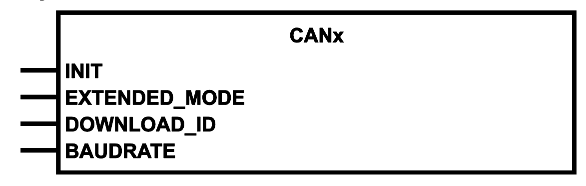
2159

x = 1...n = Nummer der CAN-Schnittstelle (je nach Gerät, → Datenblatt)

Baustein-Typ = Funktionsbaustein (FB)

Baustein ist enthalten in Bibliothek `ifm_CR7132_Vxxyyzz.LIB`

Symbol in CODESYS:



Beschreibung

2162

CANx initialisiert die x. CAN-Schnittstelle.
(x = 1...n = Nummer der CAN-Schnittstelle (je nach Gerät, → Datenblatt))

Der Download-ID muss für jede Schnittstelle unterschiedlich sein.

Die Baudraten der einzelnen CANx können unterschiedlich eingestellt werden.

► Den Eingang INIT nur für einen Zyklus bei Neustart oder Restart der Schnittstelle setzen!

> ! Eine Änderung des Download-ID und/oder der Baudrate wird erst gültig nach Spannung Aus/Ein.

Wenn der FB nicht ausgeführt wird, arbeitet die Schnittstelle mit 11-Bit-Identifier.

Parameter der Eingänge

2163

Parameter	Datentyp	Beschreibung
INIT	BOOL	TRUE (im 1. Zyklus): Baustein wird initialisiert FALSE: im weiteren Programmablauf
EXTENDED_MODE	BOOL := FALSE	TRUE: Identifier der CAN-Schnittstelle arbeitet mit 29 Bits FALSE: Identifier der CAN-Schnittstelle arbeitet mit 11 Bits
DOWNLOAD_ID	BYTE	Download-ID der CAN-Schnittstelle x x = 1...n = Nummer der CAN-Schnittstelle (je nach Gerät, → Datenblatt) zulässig = 1...127 voreingestellt = 127 - (x-1)
BAUDRATE	WORD := 125	Baudrate [kBit/s] zulässig = 20, 50, 100, 125, 250, 500, 1000

CANx_BAUDRATE

11834

x = 1...n = Nummer der CAN-Schnittstelle (je nach Gerät, → Datenblatt)

Baustein-Typ = Funktionsbaustein (FB)

Baustein ist enthalten in Bibliothek `ifm_CR7132_Vxxyyzz.LIB`

Symbol in CODESYS:



Beschreibung

11839

CANx_BAUDRATE stellt die Übertragungsrate für den Busteilnehmer ein.

Mit dem FB wird für das Gerät die Übertragungsrate eingestellt. Dazu wird am Eingang BAUDRATE der entsprechende Wert in kBit/s angegeben. Nach Ausführen des FB wird der neue Wert im Gerät gespeichert und steht auch nach einem Spannungsausfall wieder zur Verfügung.

> ! Der neue Wert wird erst nach einem RESET gültig (Spannung Aus/Ein oder Soft-Reset).

Parameter der Eingänge

655

Parameter	Datentyp	Beschreibung
ENABLE	BOOL	TRUE (im 1. Zyklus): Parameter übernehmen und aktivieren sonst: diese Funktion wird nicht ausgeführt
BAUDRATE	WORD := 125	Baudrate [kBit/s] zulässig = 20, 50, 100, 125, 250, 500, 1000

CANx_BUSLOAD

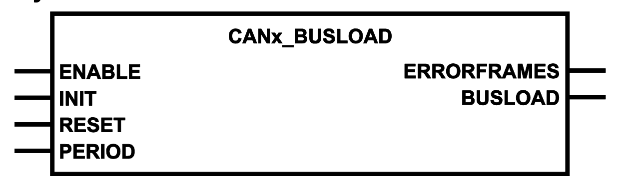
2178

x = 1...n = Nummer der CAN-Schnittstelle (je nach Gerät, → Datenblatt)

Baustein-Typ = Funktionsbaustein (FB)

Baustein ist enthalten in Bibliothek `ifm_CR7132_Vxxyyzz.LIB`

Symbol in CODESYS:



Beschreibung

2180

Ermittelt die aktuelle Buslast auf dem CAN-Bus und zählt die aufgetretenen Error-Frames.

CANx_BUSLOAD ermittelt die Buslast über die Anzahl und Länge der während der Zeit PERIOD über den CAN-Bus übertragenen Telegramme, bei Berücksichtigung der aktuellen Baudrate. Der Wert BUSLOAD wird jeweils nach Ablauf der Zeit PERIOD aktualisiert.

Ist das Bit RESET dauerhaft FALSE, wird die Anzahl der Error-Frames angezeigt, die während des letzten Zyklus aufgetreten sind.

HINWEIS

Läuft die Kommunikation auf dem CAN-Bus über das CANopen-Protokoll, dann ist es sinnvoll, den Wert von PERIOD auf die Dauer des SYNC-Zyklus zu setzen.

Die Messperiode ist dabei nicht mit dem CANopen SYNC-Zyklus synchronisiert.

Parameter der Eingänge

2181

Parameter	Datentyp	Beschreibung
ENABLE	BOOL	TRUE: Baustein ausführen FALSE: Baustein wird nicht ausgeführt > Baustein-Eingänge sind nicht aktiv > Baustein-Ausgänge sind nicht spezifiziert
INIT	BOOL	TRUE (nur 1 Zyklus lang): Konfiguration der Messdauer PERIOD FALSE: im weiteren Programmablauf
RESET	BOOL	TRUE: ERRORFRAME zurücksetzen auf "0" FALSE: Funktion wird nicht ausgeführt
PERIOD	WORD	Zeit in [ms], über welche die Buslast ermittelt wird zulässig = 20...1 000 ms

Parameter der Ausgänge

2182

Parameter	Datentyp	Beschreibung
ERRORFRAMES	WORD	Anzahl der auf dem CAN-Bus aufgetretenen Error-Frames seit dem letzten Reset
BUSLOAD	BYTE	mittlere Buslast in [%] zulässig: 0...100

CANx_DOWNLOADID

11841

= CANx Download-ID
x = 1...n = Nummer der CAN-Schnittstelle (je nach Gerät, → Datenblatt)
Baustein-Typ = Funktionsbaustein (FB)
Baustein ist enthalten in Bibliothek `ifm_CR7132_Vxxyyzz.LIB`

Symbol in CODESYS:



Beschreibung

11846

CANx_DOWNLOADID stellt den Download-Identifier für die CAN-Schnittstelle x ein.
Mit dem FB kann der Kommunikations-Identifier für den Programm-Download und das Debuggen eingestellt werden. Der neue Wert wird eingetragen, wenn der Eingang ENABLE auf TRUE gesetzt wird.

Der neue Wert wird erst nach einem RESET gültig (Spannung Aus/Ein oder Soft-Reset).

Parameter der Eingänge

649

Parameter	Datentyp	Beschreibung
ENABLE	BOOL	TRUE (im 1. Zyklus): Parameter übernehmen und aktivieren sonst: diese Funktion wird nicht ausgeführt
ID	BYTE	Download-ID der CAN-Schnittstelle x setzen x = 1...n = Nummer der CAN-Schnittstelle (je nach Gerät, → Datenblatt) zulässig = 1...127 voreingestellt = 127 - (x-1)

CANx_ERRORHANDLER

2174

x = 1...n = Nummer der CAN-Schnittstelle (je nach Gerät, → Datenblatt)

Baustein-Typ = Funktionsbaustein (FB)

Baustein ist enthalten in Bibliothek `ifm_CR7132_Vxxyyzz.LIB`

Symbol in CODESYS:

CANx_ERRORHANDLER
BUSOFF_RECOVER

Beschreibung

13990
13991

Wenn die automatische Bus-Recover-Funktion genutzt werden soll (Voreinstellung), darf CANx_ERRORHANDLER **nicht** in das Programm eingebunden und instanziert werden!

CANx_ERRORHANDLER führt ein "manuelles" Bus-Recover auf der CAN-Schnittstelle x durch.

► Nach einem erkannten CAN-Busoff den FB für einen Zyklus mit BUSOFF_RECOVER = TRUE aufrufen, damit die Steuerung wieder auf dem CAN-Bus senden und empfangen kann.

► Anschließend im Anwendungsprogramm den entsprechenden Fehler-Code zurücksetzen.

> Die CAN-Schnittstelle arbeitet wieder.

Parameter der Eingänge

2177

Parameter	Datentyp	Beschreibung
BUSOFF_RECOVER	BOOL	TRUE (nur 1 Zyklus lang): > Bus-off-Zustand beheben > Neustart der CAN-Schnittstelle x x = 1...n = Nummer der CAN-Schnittstelle (je nach Gerät, → Datenblatt) FALSE: Funktion wird nicht ausgeführt

CANx_RECEIVE

12835

x = 1...n = Nummer der CAN-Schnittstelle (je nach Gerät, → Datenblatt)

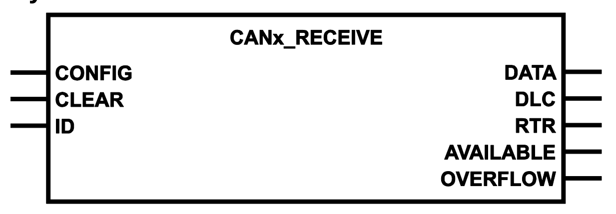
Baustein-Typ = Funktionsbaustein (FB)

Baustein ist enthalten in Bibliothek `ifm_CR7132_Vxxyyzz.LIB`

Baustein ist NICHT für Sicherheitssignale zugelassen!

► Für Sicherheitssignale folgenden FB einsetzen: → ***CAN_SAFETY_RECEIVE*** (→ Seite 214)

Symbol in CODESYS:



Beschreibung

13338

CANx_RECEIVE konfiguriert ein Datenempfangsobjekt und liest den Empfangspuffer des Datenobjektes aus.

► Den FB für jedes Datenobjekt in der Initialisierungsphase einmalig aufrufen, um dem CAN-Controller die Identifier der Datenobjekte bekannt zu machen.

► Im weiteren Programmzyklus CANx_RECEIVE zum Auslesen des jeweiligen Empfangspuffers aufrufen, bei langen Programmzyklen auch mehrfach.

► Den Ausgang AVAILABLE auswerten, so dass neu eingegangene Datenobjekte rechtzeitig aus dem Puffer gelesen und weiterverarbeitet werden.
Receive-Puffer: max. 16 Software-Puffer pro Identifier.

> Jeder Aufruf des FB dekrementiert das Byte AVAILABLE um 1.
Ist AVAILABLE = 0, sind keine Daten im Puffer.

► Den Ausgang OVERFLOW auswerten, um einen Überlauf des Datenpuffers zu erkennen.
Wenn OVERFLOW = TRUE, dann ist mindestens 1 Datenobjekt verloren gegangen.

Parameter der Eingänge

2172

Parameter	Datentyp	Beschreibung
CONFIG	BOOL	TRUE (im 1. Zyklus): Datenobjekt konfigurieren FALSE: im weiteren Programmablauf
CLEAR	BOOL	TRUE: Empfangspuffer löschen FALSE: Funktion wird nicht ausgeführt
ID	DWORD	Nummer des Datenobjekt-Identifiers: Normal Frame (2^{11} IDs): 0...2 047 = 0x0000 0000...0x0000 07FF Extended Frame (2^{29} IDs): 0...536 870 911 = 0x0000 0000...0x1FFF FFFF

Parameter der Ausgänge

632

Parameter	Datentyp	Beschreibung
DATA	ARRAY [0..7] OF BYTE	empfangene Daten (1...8 Bytes)
DLC	BYTE	Anzahl der mit SRDO empfangenen Bytes im Array DATA zulässig: 0...8
RTR	BOOL	empfangene Nachricht war ein **R**emote **T**ransmission **R**equest
AVAILABLE	BYTE	Anzahl der verbleibenden Datenbytes zulässig = 0...16 0 = keine gültigen Daten vorhanden
OVERFLOW	BOOL	TRUE: Überlauf des Datenpuffers ⇨ Datenverlust! FALSE: Datenpuffer ist ohne Datenverlust

CANx_TRANSMIT

12840

x = 1...n = Nummer der CAN-Schnittstelle (je nach Gerät, → Datenblatt)

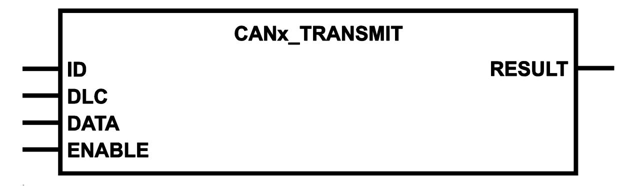
Baustein-Typ = Funktionsbaustein (FB)

Baustein ist enthalten in Bibliothek `ifm_CR7132_Vxxyyzz.LIB`

Baustein ist NICHT für Sicherheitssignale zugelassen!

► Für Sicherheitssignale folgenden FB einsetzen: → *CAN_SAFETY_TRANSMIT* (→ Seite 217)

Symbol in CODESYS:



Beschreibung

2166

CANx_TRANSMIT übergibt in jedem Aufruf ein CAN-Datenobjekt (Message) an den CAN-Controller zur Übertragung.

► Den FB für jedes Datenobjekt im Programmzyklus aufgerufen, bei langen Programmzyklen auch mehrfach.
Transmit-Puffer: max. 16 Software- und 1 Hardware-Puffer für alle Identifier zusammen.

► Den Ausgang RESULT auswerten zur Prüfung, dass der Sendeauftrag angenommen wurde.

Vereinfacht gilt bei 125 kBit/s, dass pro 1 ms ein Sendeauftrag ausgeführt werden kann.

Über den Eingang ENABLE kann die Ausführung des FB zeitweilig gesperrt werden (ENABLE = FALSE). Damit kann z.B. eine Busüberlastung verhindert werden.

Mehrere Datenobjekte können quasi gleichzeitig verschickt werden, wenn jedem Datenobjekt ein Merkerflag zugeordnet wird und mit diesem die Ausführung des FB über den ENABLE-Eingang gesteuert wird.

Parameter der Eingänge

2167

Parameter	Datentyp	Beschreibung
ID	DWORD	Nummer des Datenobjekt-Identifiers: Normal Frame (2^{11} IDs): 0...2 047 = 0x0000 0000...0x0000 07FF Extended Frame (2^{29} IDs): 0...536 870 911 = 0x0000 0000...0x1FFF FFFF
DLC	BYTE	Anzahl der mit SRDO empfangenen Bytes im Array DATA zulässig: 0...8
DATA	ARRAY [0..7] OF BYTE	zu sendende Daten (1...8 Bytes)
ENABLE	BOOL	TRUE: Baustein ausführen FALSE: Baustein wird nicht ausgeführt > Baustein-Eingänge sind nicht aktiv > Baustein-Ausgänge sind nicht spezifiziert

Parameter der Ausgänge

2168

Parameter	Datentyp	Beschreibung
RESULT	BOOL	TRUE (nur 1 Zyklus lang): der Baustein hat den Sendeauftrag angenommen FALSE: Sendeauftrag wurde nicht angenommen

6.2.3 Bausteine: Daten sicher übertragen

Inhalt

13756

Für Sicherheitsfunktionen der SafetyController stellen wir folgende zertifizierte CAN-Bausteine zur Verfügung:

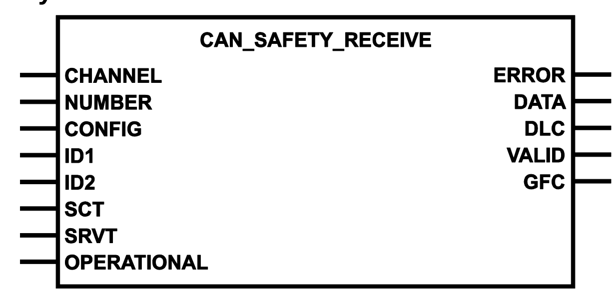
CAN_SAFETY_RECEIVE

12848

Baustein-Typ = Funktionsbaustein (FB)

Baustein ist enthalten in Bibliothek `ifm_CR7132_Vxxyyzz.LIB`

Symbol in CODESYS:



Beschreibung

12850

CANsafety-FBs benötigen 2 mit 11 Bit betriebene CAN-Kanäle gleichzeitig. Der Extended-Modus ist für mit CANsafety genutzte Schnittstellen nicht zulässig!

CAN_SAFETY_RECEIVE empfängt eine sichere CAN-Nachricht (SRDO).
Der SafetyController unterstützt für beide CANsafety-Schnittstellen zusammen bis zu 8 TX-SRDOs und 8 RX-SRDOs.

Der FB initialisiert, konfiguriert und empfängt ein SRDO. Der FB muss in folgender Reihenfolge verwendet werden:

► Die mit CHANNEL, NUMBER, ID1, ID2, SCT und SRVT eingestellte Konfiguration mit CONFIG = TRUE an den Baustein übergeben.

► Kommunikation starten mit...
  • OPERATIONAL = TRUE
  • **und** CONFIG = FALSE.

> Die Konfiguration wird fixiert und durch eine Prüfsumme gesichert.

> Empfangene Daten werden im Array abgelegt und der Ausgang VALID für einen Zyklus auf TRUE gesetzt.

► Im Anwendungsprogramm müssen die Daten gelesen und sicher ausgewertet werden.

> Werden die Daten fehlerhaft oder außerhalb der festgelegten Zeitgrenzen empfangen, geht die Steuerung in den sicheren Zustand (alle Ausgänge aus).

> Jede Millisekunde wird 1 Sende-SRDO und 1 Empfangs-SRDO bearbeitet.
Das bedeutet: Ist nur 1 SRDO im Programm definiert, kann es jede ms übertragen werden. Bei 8 SRDOs wird jedes Objekt nur alle 8 ms bearbeitet.

Je nach Buslast (weitere CAN-Nachrichten unabhängig von CANsafety) muss bei 8 Sende-SRDOs eine SRVT von 16...24 ms eingestellt werden. Bei sehr hoher Buslast vergrößert sich die Zeit zwischen den normalen und invertierten Datentelegrammen, daraus resultiert eine noch längere SRVT.

Einstellempfehlung bei sehr hoher Buslast durch CANopen und 8 SRDOs:
• REFRESHTIME = 100 ms
• SCT = 150 ms
• SRVT = 40 ms

Die SCT wirkt sich auf die Sicherheitszeit im Gesamtsystem aus.
► Auswirkung der SCT auf die Sicherheitszeit im Projekt berücksichtigen!

HINWEIS

Bevor der FB aktiviert wird (OPERATIONAL = TRUE), müssen gültige CANsafety-Daten (richtiger Identifier, richtige Reihenfolge usw.) auf dem Bus übertragen werden.

Andernfalls wird die Fehlerüberwachung des FB aktiv, der ERROR-Ausgang gesetzt und die Steuerung geht in den sicheren Zustand. Es ist dann keine sichere CAN-Kommunikation mehr möglich.

Parameter der Eingänge

12851

Parameter	Datentyp	Beschreibung
CHANNEL	BYTE	CANsafety-Kanal (Schnittstellenpaar) 01 = CAN-Schnittstellen 1 + 2 02 = CAN-Schnittstellen 3 + 4 CANsafety-FBs benötigen 2 mit 11 Bit betriebene CAN-Kanäle gleichzeitig. Der Extended-Modus ist für mit CANsafety genutzte Schnittstellen nicht zulässig!
NUMBER	BYTE	Nummer des SRDOs zulässig = 0...7 Für jede FB-Instanz einen anderen Wert verwenden!
CONFIG	BOOL	Bevor der FB für den operativen Betrieb aktiviert werden darf (OPERATIONAL=TRUE), eine gültige Konfiguration vornehmen! Voraussetzung: OPERATIONAL=FALSE TRUE (nur 1 Zyklus lang): Konfigurationswerte an den Eingängen übernehmen FALSE: im weiteren Programmablauf
ID1	DWORD	SRDO-CAN-ID für das CAN-Telegramm mit den Originaldaten Die CAN-ID muss einen ungeraden Wert haben! empfohlen: 101_{16}, 103_{16}, 105_{16}, ... , $17D_{16}$, $17F_{16}$
ID2	DWORD	SRDO-CAN-ID für das CAN-Telegramm mit den invertierten Daten Vorgabe: ID2 = ID1 + 1
SCT	TIME	SCT = Sicherheits-Zykluszeit: maximale Zeit, in der das SRDO zyklisch empfangen werden muss empfohlen: ≥ 10 ms Die SCT im empfangenden Gerät muss länger sein als im sendenden Gerät die REFRESHTIME für ***CAN_SAFETY_TRANSMIT*** (→ Seite 217). Wird die SCT überschritten, wird der Merker ERROR auf TRUE gesetzt und die Steuerung geht in den "sicheren Zustand".
SRVT	TIME	SRVT = sicherheitsrelevante Objekt-Gültigkeitsdauer: maximal zulässige Zeit zwischen dem CAN-Telegramm mit den Originaldaten und dem CAN-Telegramm mit den invertierten Daten. empfohlen: ≥ 5 ms Wird die SRVT überschritten, wird der Merker ERROR auf TRUE gesetzt und die Steuerung geht in den "sicheren Zustand".
OPERATIONAL	BOOL	TRUE: das SRDO wird zyklisch und sicher übertragen der Baustein kann NICHT umkonfiguriert werden FALSE: das SRDO wird nicht übertragen der Baustein kann konfiguriert werden

Parameter der Ausgänge

12852

Parameter	Datentyp	Beschreibung
ERROR	BYTE	Fehler-Code nach Empfangen eines SRDOs: 0 = 0x00 kein Fehler 2 = 0x02 Datenflusskontrolle in 1. CANsafety-Schnittstelle 3 = 0x03 Datenflusskontrolle in 2. CANsafety-Schnittstelle 4 = 0x04 falsche Anzahl an Daten empfangen 5 = 0x05 unterschiedliche Daten im SRDO empfangen 6 = 0x06 SCT überschritten 7 = 0x07 SRVT überschritten 8 = 0x08 unerwarteter Datenempfang auf der 1. CANsafety-Schnittstelle 9 = 0x09 unerwarteter Datenempfang auf der 2. CANsafety-Schnittstelle 10 = 0x0A CRC-Fehler Konfigurationsdaten 11 = 0x0B fehlerhaftes Identifier-Paar (Konfiguration) 12 = 0x0C SRDO-Nummer ungültig (Konfiguration) 13 = 0x0D unzulässiger Kanal (Konfiguration) 14 = 0x0E Konfigurationsversuch im Operational-Modus
DATA	ARRAY [0..7] OF BYTE	empfangene Daten (1...8 Bytes)
DLC	BYTE	Anzahl der mit SRDO empfangenen Bytes im Array DATA zulässig: 0...8
VALID	BOOL	TRUE (nur 1 Zyklus lang): neue gültige Daten wurden empfangen FALSE: im weiteren Programmablauf
GFC	BOOL	TRUE (nur 1 Zyklus lang): Baustein empfing das "**G**lobal **f**ailsafe **c**ommand" FALSE: im weiteren Programmablauf

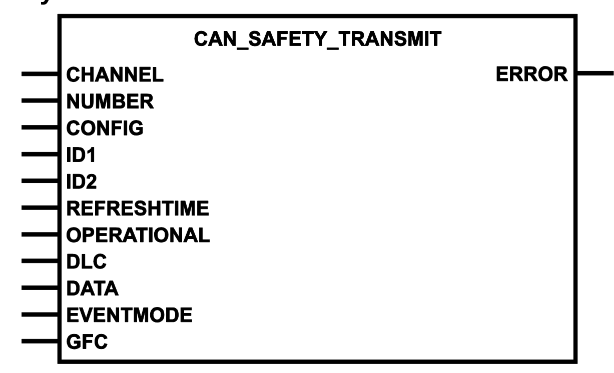
CAN_SAFETY_TRANSMIT

12865

Baustein-Typ = Funktionsbaustein (FB)

Baustein ist enthalten in Bibliothek `ifm_CR7132_Vxxyyzz.LIB`

Symbol in CODESYS:



Beschreibung

12867

> ❗ CANsafety-FBs benötigen 2 mit 11 Bit betriebene CAN-Kanäle gleichzeitig. Der Extended-Modus ist für mit CANsafety genutzte Schnittstellen nicht zulässig!

CAN_SAFETY_TRANSMIT überträgt eine sichere CAN-Nachricht (SRDO).
Der SafetyController unterstützt für beide CANsafety-Schnittstellen zusammen bis zu 8 TX-SRDOs und 8 RX-SRDOs.

Der FB initialisiert, konfiguriert und überträgt ein SRDO. Der FB muss in folgender Reihenfolge verwendet werden:

► Die mit CHANNEL, NUMBER, ID1, ID2 und REFRESHTIME eingestellte Konfiguration mit CONFIG = TRUE an den FB übergeben.

► Kommunikation starten mit...
 • OPERATIONAL = TRUE
 • **und** CONFIG = FALSE.

> Die Konfiguration wird fixiert und durch eine Prüfsumme gesichert.

> Jede Millisekunde wird 1 Sende-SRDO und 1 Empfangs-SRDO bearbeitet.
Das bedeutet: Ist nur 1 SRDO im Programm definiert, kann es jede ms übertragen werden. Bei 8 SRDOs wird jedes Objekt nur alle 8 ms bearbeitet.

Je nach Buslast (weitere CAN-Nachrichten unabhängig von CANsafety) muss bei 8 Sende-SRDOs eine SRVT von 16...24 ms eingestellt werden. Bei sehr hoher Buslast vergrößert sich die Zeit zwischen den normalen und den invertierten Datentelegrammen, daraus resultiert eine noch längere SRVT.

Einstellempfehlung bei sehr hoher Buslast durch CANopen und 8 SRDOs:
• REFRESHTIME = 100 ms
• SCT = 150 ms
• SRVT = 40 ms

Übertragung von analogen Daten über ein SRDO:
► Den Eingang EVENTMODE = FALSE setzen.
Andernfalls kann es zu einer extremen Belastung des CAN-Bus kommen.

Parameter der Eingänge

12868

Parameter	Datentyp	Beschreibung
CHANNEL	BYTE	CANsafety-Kanal (Schnittstellenpaar) 01 = CAN-Schnittstellen 1 + 2 02 = CAN-Schnittstellen 3 + 4 (!) CANsafety-FBs benötigen 2 mit 11 Bit betriebene CAN-Kanäle gleichzeitig. Der Extended-Modus ist für mit CANsafety genutzte Schnittstellen nicht zulässig!
NUMBER	BYTE	Nummer des SRDOs zulässig = 0...7 (!) Für jede FB-Instanz einen anderen Wert verwenden!
CONFIG	BOOL	Bevor der FB für den operativen Betrieb aktiviert werden darf (OPERATIONAL=TRUE), eine gültige Konfiguration vornehmen! Voraussetzung: OPERATIONAL=FALSE TRUE (nur 1 Zyklus lang): Konfigurationswerte an den Eingängen übernehmen FALSE: im weiteren Programmablauf
ID1	DWORD	SRDO-CAN-ID für das CAN-Telegramm mit den Originaldaten (!) Die CAN-ID muss einen ungeraden Wert haben! empfohlen: 101_{16}, 103_{16}, 105_{16}, ... , $17D_{16}$, $17F_{16}$
ID2	DWORD	SRDO-CAN-ID für das CAN-Telegramm mit den invertierten Daten Vorgabe: ID2 = ID1 + 1
REFRESHTIME	TIME	Zeit, in der das SRDO spätestens gesendet wird empfohlen: ≥ 10 ms Die REFRESHTIME im sendenden Gerät muss kürzer sein als im empfangenden Gerät die SCT für ***CAN_SAFETY_RECEIVE*** (→ Seite 214).
OPERATIONAL	BOOL	TRUE: das SRDO wird zyklisch und sicher übertragen der Baustein kann NICHT umkonfiguriert werden FALSE: das SRDO wird nicht übertragen der Baustein kann konfiguriert werden
DLC	BYTE	Anzahl der mit SRDO zu übertragenden Bytes aus dem Array DATA zulässig: 0...8
DATA	ARRAY [0..7] OF BYTE	zu sendende Daten (1...8 Bytes)
EVENTMODE	BOOL	TRUE: Daten werden ereignisgesteuert übertragen: • sofort bei Änderung im DATA-Array • spätestens nach Ablauf von REFRESHTIME FALSE: Daten werden übertragen: zyklisch im Abstand von REFRESHTIME
GFC	BOOL	FALSE ⇨ TRUE (Flanke): "**G**lobal **f**ailsafe **c**ommand" einmalig senden sonst: diese Funktion wird nicht ausgeführt

Parameter der Ausgänge

12869

Parameter	Datentyp	Beschreibung
ERROR	BYTE	Fehler-Code nach Senden eines SRDOs: 0 = 0x00 kein Fehler 1 = 0x01 Transmit Timeout 10 = 0x0A CRC-Fehler Konfigurationsdaten 11 = 0x0B fehlerhaftes Identifier-Paar (Konfiguration) 12 = 0x0C SRDO-Nummer ungültig (Konfiguration) 13 = 0x0D unzulässiger Kanal (Konfiguration) 14 = 0x0E Konfigurationsversuch im Operational-Modus

6.2.4 Bausteine: CANopen-Master

Inhalt

1870

Für den CANopen-Master stellt **ifm electronic** eine Reihe von Bausteinen zur Verfügung, die im Folgenden erklärt werden.

CANx_MASTER_EMCY_HANDLER

2006

x = 1...n = Nummer der CAN-Schnittstelle (je nach Gerät, → Datenblatt)

Baustein-Typ = Funktionsbaustein (FB)

Baustein ist enthalten in Bibliothek `ifm_CR7132_CANopenxMaster_Vxxyyzz.LIB`

Symbol in CODESYS:



Beschreibung

2009

CANx_MASTER_EMCY_HANDLER verwaltet den geräteeigenen Fehlerstatus des Masters. Der FB muss in folgenden Fällen aufgerufen werden:

- der Fehlerstatus soll ins Netzwerk übertragen werden und
- die Fehlermeldungen des Anwendungsprogramms sollen im Objektverzeichnis gespeichert werden.

Über den FB können die aktuellen Werte aus dem Error-Register (Index 0x1001/01) und Error Field (Index 0x1003/0-5) des CANopen-Objektverzeichnis ausgelesen werden.

Sollen anwendungsspezifische Fehlernachrichten im Objektverzeichnis gespeichert werden, muss CANx_MASTER_EMCY_HANDLER **nach** dem (mehrfachen) Bearbeiten von ***CANx_MASTER_SEND_EMERGENCY*** (→ Seite 222) aufgerufen werden.

Parameter der Eingänge

2010

Parameter	Datentyp	Beschreibung
CLEAR_ERROR_FIELD	BOOL	FALSE ⇨ TRUE (Flanke): • Inhalt des ERROR_FIELD an FB-Ausgang ausgeben • Inhalt des ERROR_FIELD im Objektverzeichnis löschen sonst: diese Funktion wird nicht ausgeführt

Parameter der Ausgänge

2011

Parameter	Datentyp	Beschreibung
ERROR_REGISTER	BYTE	Zeigt den Inhalt des OBV Index 0x1001 (Error-Register) → Kapitel ***Objekt 0x1001 (Error-Register)*** (→ Seite 416)
ERROR_FIELD	ARRAY [0..5] OF WORD	Zeigt den Inhalt des OBV Index 0x1003 (Error-Field) → Kapitel ***Objekt 0x1003 (Error Field)*** (→ Seite 413) ERROR_FIELD[0]: Anzahl der gespeicherten Fehler ERROR_FIELD[1...5]: gespeicherte Fehler, der jüngste Fehler steht im Index [1]

CANx_MASTER_SEND_EMERGENCY

2012

x = 1...n = Nummer der CAN-Schnittstelle (je nach Gerät, → Datenblatt)

Baustein-Typ = Funktionsbaustein (FB)

Baustein ist enthalten in Bibliothek `ifm_CR7132_CANopenxMaster_Vxxyyzz.LIB`

Symbol in CODESYS:

```
CANx_MASTER_SEND_EMERGENCY
ENABLE
ERROR
ERROR_CODE
ERROR_REGISTER
MANUFACTURER_ERROR_FIELD
```

Beschreibung

2015

CANx_MASTER_SEND_EMERGENCY versendet anwendungsspezifische Fehlerstatus. Der FB wird aufgerufen, wenn der Fehlerstatus an andere Geräte im Netzwerkverbund übertragen werden soll.

> ! Sollen anwendungsspezifische Fehlernachrichten im Objektverzeichnis gespeichert werden, muss ***CANx_MASTER_EMCY_HANDLER*** (→ Seite 221) **nach** dem (mehrfachen) Bearbeiten von CANx_MASTER_SEND_EMERGENCY aufgerufen werden.

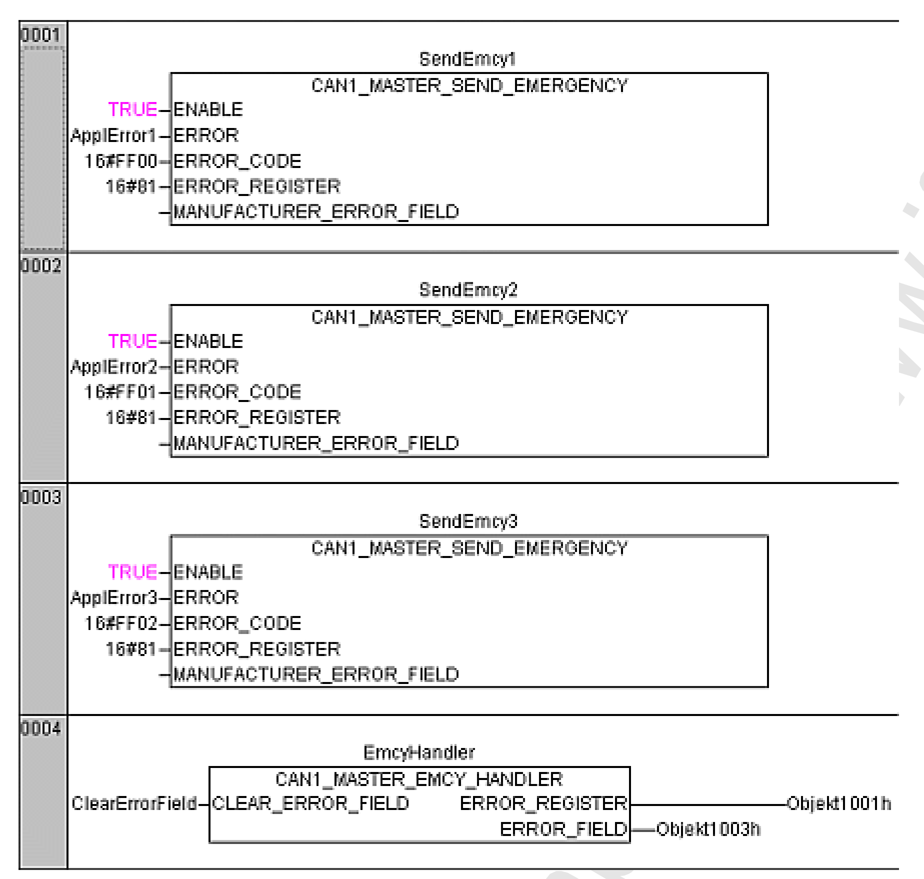
Parameter der Eingänge

2016

Parameter	Datentyp	Beschreibung
ENABLE	BOOL	TRUE: Baustein ausführen FALSE: Baustein wird nicht ausgeführt > Baustein-Eingänge sind nicht aktiv > Baustein-Ausgänge sind nicht spezifiziert
ERROR	BOOL	FALSE ⇨ TRUE (Flanke): sendet den anstehenden Fehler-Code TRUE ⇨ FALSE (Flanke) UND Fehler steht nicht mehr an: Nach Verzögerung von ca. 1 s: > Null-Fehlermeldung wird gesendet sonst: diese Funktion wird nicht ausgeführt
ERROR_CODE	WORD	Der Error-Code gibt detailliert Auskunft über den erkannten Fehler. Die Werte sollten gemäß der CANopen-Spezifikation eingetragen werden. → Kapitel ***Übersicht CANopen Error-Codes*** (→ Seite 415)
ERROR_REGISTER	BYTE	ERROR_REGISTER gibt die Art des Fehlers an. Der hier angegebene Wert wird mit allen anderen aktuell aktiven Fehlernachrichten bitweise ODER-verknüpft. Der sich hierbei ergebende Wert wird ins Error-Register (Index 1001_{16}/00) geschrieben und mit der EMCY-Nachricht versendet. Die Werte sollten gemäß der CANopen-Spezifikation eingetragen werden.
MANUFACTURER_ERROR_FIELD	ARRAY [0..4] OF BYTE	Hier können bis zu 5 Bytes anwendungsspezifische Fehlerinformationen eingetragen werden. Das Format ist dabei frei wählbar.

Beispiel: CANx_MASTER_SEND_EMERGENCY

2018



In diesem Beispiel werden nacheinander 3 Fehlermeldungen generiert:

1. ApplError1, Code = 0xFF00 im Fehlerregister 0x81
2. ApplError2, Code = 0xFF01 im Fehlerregister 0x81
3. ApplError3, Code = 0xFF02 im Fehlerregister 0x81

Der FB CAN1_MASTER_EMCY_HANDLER sendet die Fehlermeldungen an das Fehler-Register "Objekt 0x1001" im Fehler-Array "Objekt 0x1003".

CANx_MASTER_STATUS

2692

x = 1...n = Nummer der CAN-Schnittstelle (je nach Gerät, → Datenblatt)

Baustein-Typ = Funktionsbaustein (FB)

Baustein ist enthalten in Bibliothek `ifm_CR7132_CANopenxMaster_Vxxyyzz.LIB`

Symbol in CODESYS:

CANx_MASTER_STATUS	
GLOBAL_START	NODE_ID
CLEAR_RX_OVERFLOW_FLAG	BAUDRATE
CLEAR_RX_BUFFER	NODE_STATE
CLEAR_TX_OVERFLOW_FLAG	SYNC
CLEAR_TX_BUFFER	RX_OVERFLOW
CLEAR_OD_CHANGED_FLAG	TX_OVERFLOW
CLEAR_ERROR_CONTROL	OD_CHANGED
RESET_ALL_NODES	ERROR_CONTROL
START_ALL_NODES	GET_EMERGENCY
NODE_STATE_SLAVE	FIRST_NODE_INDEX
EMERGENCY_OBJECT_SLAVES	LAST_NODE_INDEX

Beschreibung

2024

Status-Anzeige des als CANopen-Master eingesetzten Gerätes

Der FB zeigt den Status des als CANopen-Master eingesetzten Gerätes an. Weitere Möglichkeiten:
- den Status des Netzwerks überwachen
- den Status der angeschlossenen Slaves überwachen
- die Slaves im Netzwerk zurücksetzen oder starten.

Der FB vereinfacht die Anwendung der CODESYS-CANopen-Master-Bibliotheken. Wir empfehlen dringend, die Auswertung des Netzwerkstatus und der Fehlermeldungen über diesen FB vorzunehmen.

Parameter der Eingänge

2695

Parameter	Datentyp	Beschreibung
GLOBAL_START	BOOL	TRUE: Alle angeschlossenen Netzwerkteilnehmer (Slaves) werden gleichzeitig bei der Netzwerkinitialisierung gestartet (⇨ Zustand OPERATIONAL). FALSE: Die angeschlossenen Netzwerkteilnehmer werden einzeln nacheinander gestartet.
CLEAR_RX_OVERFLOW_FLAG	BOOL	FALSE ⇨ TRUE (Flanke): Fehlerflag RX_OVERFLOW löschen sonst: diese Funktion wird nicht ausgeführt
CLEAR_RX_BUFFER	BOOL	FALSE ⇨ TRUE (Flanke): Daten im Empfangspuffer löschen sonst: diese Funktion wird nicht ausgeführt
CLEAR_TX_OVERFLOW_FLAG	BOOL	FALSE ⇨ TRUE (Flanke): Fehlerflag TX_OVERFLOW löschen sonst: diese Funktion wird nicht ausgeführt
CLEAR_TX_BUFFER	BOOL	FALSE ⇨ TRUE (Flanke): Daten im Sendepuffer löschen sonst: diese Funktion wird nicht ausgeführt
CLEAR_OD_CHANGED_FLAG	BOOL	FALSE ⇨ TRUE (Flanke): Flag OD_CHANGED löschen sonst: diese Funktion wird nicht ausgeführt
CLEAR_ERROR_CONTROL	BOOL	FALSE ⇨ TRUE (Flanke): Die Guard-Fehlerliste (ERROR_CONTROL) löschen sonst: diese Funktion wird nicht ausgeführt
RESET_ALL_NODES	BOOL	FALSE ⇨ TRUE (Flanke): Alle angeschlossenen Netzwerkteilnehmer (Slaves) werden per NMT-Kommando zurückgesetzt sonst: diese Funktion wird nicht ausgeführt
START_ALL_NODES	BOOL	FALSE ⇨ TRUE (Flanke): Alle angeschlossenen Netzwerkteilnehmer (Slaves) werden per NMT-Kommando gestartet sonst: diese Funktion wird nicht ausgeführt
NODE_STATE_SLAVES	DWORD	Zeigt den Status aller Netzwerkknoten. Beispiel-Code → Kapitel ***Beispiel: CANx_MASTER_STATUS*** (→ Seite 228)
EMERGENCY_OBJECT_SLAVES	DWORD	Zeigt die zuletzt aufgetretenen Fehlermeldungen aller Netzwerkknoten. → Kapitel ***Zugriff auf die Strukturen zur Laufzeit der Anwendung*** (→ Seite 229)

Parameter der Ausgänge

2696

Parameter	Datentyp	Beschreibung
NODE_ID	BYTE	aktueller Knoten-ID
BAUDRATE	WORD	aktuelle Baudrate des Knotens in [kBaud]
NODE_STATE	INT	aktueller Status des CANopen-Masters
SYNC	BOOL	SYNC-Signal des CANopen-Masters TRUE: Im letzten Zyklus wurde ein SYNC-Signal gesendet FALSE: Im letzten Zyklus wurde kein SYNC-Signal gesendet
RX_OVERFLOW	BOOL	Fehler: Empfangspuffer-Überlauf
TX_OVERFLOW	BOOL	Fehler: Sendepuffer-Überlauf
OD_CHANGED	BOOL	Daten im Objektverzeichnis des CANopen-Masters wurden geändert
ERROR_CONTROL	ARRAY [0..7] OF BYTE	Das Array enthält die Liste (max. 8) der fehlenden Netzwerkknoten (Guard- oder Heartbeat-Fehler) → Kapitel ***Zugriff auf die Strukturen zur Laufzeit der Anwendung*** (→ Seite 229)
GET_EMERGENCY	STRUCT EMERGENY_MESSAGE	Am Ausgang stehen die Daten für die Struktur EMERGENY_MESSAGE zur Verfügung. Es wird immer die letzte Fehlermeldung eines Netzwerkknotens angezeigt. Um eine Liste aller aufgetretenen Fehler zu erhalten, muss das Array "EMERGENCY_OBJECT_SLAVES" ausgewertet werden.
FIRST_NODE_INDEX	INT	Bereich, in dem sich die Knotennummern der an diesem CAN-Bus angeschlossenen Knoten (Slaves) befinden
LAST_NODE_INDEX	INT	

Parameter der internen Strukturen

2698

Hier sehen Sie die Strukturen der in diesem Baustein genutzten Arrays.

Die Anwendung des FB CANx_MASTER_STATUS zeigen Ihnen die Code-Fragmente am Beispiel des Controllers CR0032 → Kapitel ***Beispiel: CANx_MASTER_STATUS*** (→ Seite 228).

Struktur von CANx_EMERGENGY_MESSAGE

13996

Die Struktur ist in den globalen Variablen der Bibliothek `ifm_CR7132_CANopenMaster_Vxxyyzz.LIB` angelegt.

Parameter	Datentyp	Beschreibung
NODE_ID	BYTE	Node-ID des Teilnehmers, von dem die EMCY-Nachricht empfangen wurde
ERROR_CODE	WORD	Error-Code mit der Information, welcher Fehler aufgetreten ist. → CANopen-Spezifikation CiA Draft Standard 301 Version 4
ERROR_REGISTER	BYTE	Wert im Error-Register (Index 0x1001/00) des sendenden Teilnehmers
MANUFACTURER_ERROR_FIELD	ARRAY [0..4] OF BYTE	herstellerspezifischer Datenbereich in der EMCY-Nachricht

Struktur von CANx_NODE_STATE

13997

Die Struktur ist in den globalen Variablen der Bibliothek ifm_CR7132_CANopenMaster_Vxxyyzz.LIB angelegt.

Parameter	Datentyp	Beschreibung
NODE_ID	BYTE	Node-ID des CANopen-Slaves, zu dem die Statusinformationen und Konfigurationsflags in der Struktur gehören
NODE_STATE	BYTE	aktueller Status des CANopen-Slaves aus Sicht des CANopen-Stacks des CANopen-Masters
LAST_STATE	BYTE	der letzte bekannte Status des CANopen-Slaves 0 = Bootup-Nachricht vom CANopen-Slave empfangen 4 = CANopen-Slave im Status PRE-OPERATIONAL und wird per SDO-Zugriff konfiguriert 5 = CANopen-Slave im Status OPERATIONAL 127 = CANopen-Slave im Status PRE-OPERATIONAL
RESET_NODE	BOOL	Flag zum manuellen Zurücksetzen des CANopen-Slaves (NMT-Kommando = `Reset_Node`)
START_NODE	BOOL	Flag zum manuellen Starten des CANopen-Slaves (NMT-Kommando = `start`)
PREOP_NODE	BOOL	Flag zum manuellen Versetzen des CANopen-Slaves in den Zustand PRE-OPERATIONAL (NMT-Kommando = `enter PRE-OPERATIONAL`)
SET_TIMEOUT_STATE	BOOL	Flag zum manuellen Überspringen der Initialisierung eines CANopen-Slaves, wenn Folgendes zutrifft: • Slave ist nicht im Netzwerk vorhanden • **und** Slave ist nicht als optional konfiguriert
SET_NODE_STATE	BOOL	Flag zum manuellen Einleiten der Initialisierung eines CANopen-Slaves Der Slave hatte sich beim Zugriff auf das Objekt 0x1000 als ein anderer Gerätetyp identifiert, als in der EDS-Datei angegeben ist, die in der CODESYS-Steuerungskonfiguration eingebunden wurde

Beispiel: CANx_MASTER_STATUS

2031

Slave-Informationen

2699

Damit Sie auf die Informationen der einzelnen CANopen-Knoten zugreifen können, müssen Sie ein Array der jeweiligen Struktur anlegen. Die Strukturen sind in der Bibliothek enthalten. Sie können Sie im Bibliotheksverwalter unter [Datentypen] sehen.

Die Anzahl der Array-Elemente wird bestimmt durch die Globale Variable MAX_NODEINDEX, die automatisch vom CANopen-Stack angelegt wird. Sie enthält die Anzahl der im Netzwerkkonfigurator angegebenen Slaves minus 1.

> ! Die Nummern der Array-Elemente entsprechen **nicht** der Node-ID. Der Identifier kann aus der jeweiligen Struktur unter NODE_ID ausgelesen werden.

```
PROGRAM MasterStatus
VAR
    Status: CR0032_MASTER_STATUS;
    StartAllNodes: BOOL:= TRUE;
    ClearRxOverflowFlag: BOOL;
    ClearRxBuffer: BOOL;
    ClearTxOverflowFlag: BOOL;
    ClearTxBuffer: BOOL;
    ClearOdChanged: BOOL;
    ClearErrorControl: BOOL;
    ResetAllNodes: BOOL;
    ResetSingleNodeArray: ARRAY[0..MAX_NODEINDEX] OF RESET_NODE;
    NodeStateSlavesArray: ARRAY [0..MAX_NODEINDEX] OF NODE_STATE;
    EmergencyObjectSlavesArray: ARRAY[0..MAX_NODEINDEX] OF EMERGENCY_MESSAGE;
    node_id: BYTE;
    baudrate: WORD;
    node_state: INT;
    Sync: BOOL;
    RxOverflow: BOOL;
    TxOverflow: BOOL;
    OdChanged: BOOL;
    GuardHeartbeatErrorArray: ARRAY[0..7] OF BYTE;
    GetEmergency: EMERGENCY_MESSAGE;
END_VAR
```

Struktur Knoten-Status

2034

```
TYPE CAN1_NODE_STATE :
STRUCT
    NODE_ID: BYTE;
    NODE_STATE: BYTE;
    LAST_STATE: BYTE;
    RESET_NODE: BOOL;
    START_NODE: BOOL;
    PREOP_NODE: BOOL;
    SET_TIMEOUT_STATE: BOOL;
    SET_NODE_STATE: BOOL;
END_STRUCT
END_TYPE
```

Struktur Emergency_Message

2035

```
TYPE CAN1_EMERGENCY_MESSAGE :
STRUCT
    NODE_ID: BYTE;
    ERROR_CODE: WORD;
    ERROR_REGISTER: BYTE;
    MANUFACTURER_ERROR_FIELD: ARRAY[0..4] OF BYTE;
END_STRUCT
END_TYPE
```

Zugriff auf die Strukturen zur Laufzeit der Anwendung

2036

Zur Laufzeit können Sie auf das jeweilige Array-Element über die globalen Variablen der Bibliothek zugreifen und so den Status oder die EMCY-Nachrichten auslesen oder den Knoten zurücksetzen.

```
⊟--NodeStateList
    ⊟--NodeStateList[0]
        ---.NODE_ID = 16#02
        ---.NODE_STATE = 16#04
        ---.LAST_STATE = 16#00
        ---.RESET_NODE = FALSE <:= TRUE>
        ---.START_NODE = FALSE
        ---.PREOP_NODE = FALSE
        ---.SET_TIMEOUT_STATE = FALSE
        ---.SET_NODE_STATE = FALSE
    ⊟--NodeStateList[1]
        ---.NODE_ID = 16#03
        ---.NODE_STATE = 16#03
        ---.LAST_STATE = 16#00
        ---.RESET_NODE = FALSE
        ---.START_NODE = FALSE
        ---.PREOP_NODE = FALSE
        ---.SET_TIMEOUT_STATE = FALSE
        ---.SET_NODE_STATE = FALSE
⊟--NodeEmergencyList
    ⊟--NodeEmergencyList[0]
        ---.NODE_ID = 16#02
        ---.ERROR_CODE = 16#0000
        ---.ERROR_REGISTER = 16#00
        ⊟--.MANUFACTURER_ERROR_FIELD
            ---.MANUFACTURER_ERROR_FIELD[0] = 16#00
            ---.MANUFACTURER_ERROR_FIELD[1] = 16#00
            ---.MANUFACTURER_ERROR_FIELD[2] = 16#00
            ---.MANUFACTURER_ERROR_FIELD[3] = 16#00
            ---.MANUFACTURER_ERROR_FIELD[4] = 16#00
    ⊟--NodeEmergencyList[1]
        ---.NODE_ID = 16#03
        ---.ERROR_CODE = 16#0000
        ---.ERROR_REGISTER = 16#00
        ⊟--.MANUFACTURER_ERROR_FIELD
            ---.MANUFACTURER_ERROR_FIELD[0] = 16#00
            ---.MANUFACTURER_ERROR_FIELD[1] = 16#00
            ---.MANUFACTURER_ERROR_FIELD[2] = 16#00
            ---.MANUFACTURER_ERROR_FIELD[3] = 16#00
            ---.MANUFACTURER_ERROR_FIELD[4] = 16#00
```

Setzen Sie im obigen Beispiel ResetSingleNodeArray[0].RESET_NODE kurzzeitig auf TRUE, wird der erste Knoten im Konfigurationsbaum zurückgesetzt.

🛈 zu den möglichen Fehler-Codes → Kapitel ***CAN / CANopen: Fehler und Fehlerbehandlung*** (→ Seite 409).

6.2.5 Bausteine: CANopen-Slave

Inhalt

1874

Für den CANopen-Slave stellt **ifm electronic** eine Reihe von Bausteinen zur Verfügung, die im Folgenden erklärt werden.

CANx_SLAVE_EMCY_HANDLER

2050

x = 1...n = Nummer der CAN-Schnittstelle (je nach Gerät, → Datenblatt)

Baustein-Typ = Funktionsbaustein (FB)

Baustein ist enthalten in Bibliothek `ifm_CR7132_CANopenxSlave_Vxxyyzz.LIB`

Symbol in CODESYS:



Beschreibung

2053

CANx_SLAVE_EMCY_HANDLER verwaltet den geräteeigenen Fehlerstatus des CANopen-Slaves:
- Error Register (Index 0x1001) und
- Error Field (Index 0x1003) des CANopen Objektverzeichnis.

► Den FB in folgenden Fällen aufrufen:
- der Fehlerstatus soll ins CAN-Netzwerk übertragen werden und
- die Fehlernachrichten des Anwendungsprogramms sollen im Objektverzeichnis gespeichert werden.

Sollen die Fehlernachrichten im Objektverzeichnis gespeichert werden?

► **Nach** dem (mehrfachen) Bearbeiten von ***CANx_SLAVE_SEND_EMERGENCY*** (→ Seite 233) einmalig CANx_SLAVE_EMCY_HANDLER aufrufen!

Parameter der Eingänge

2054

Parameter	Datentyp	Beschreibung
CLEAR_ERROR_FIELD	BOOL	FALSE ⇨ TRUE (Flanke): • Inhalt des ERROR_FIELD an FB-Ausgang ausgeben • Inhalt des ERROR_FIELD im Objektverzeichnis löschen sonst: diese Funktion wird nicht ausgeführt

Parameter der Ausgänge

2055

Parameter	Datentyp	Beschreibung
ERROR_REGISTER	BYTE	Zeigt den Inhalt des OBV Index 0x1001 (Error-Register) → Kapitel ***Objekt 0x1001 (Error-Register)*** (→ Seite 416)
ERROR_FIELD	ARRAY [0..5] OF WORD	Zeigt den Inhalt des OBV Index 0x1003 (Error-Field) → Kapitel ***Objekt 0x1003 (Error Field)*** (→ Seite 413) ERROR_FIELD[0]: Anzahl der gespeicherten Fehler ERROR_FIELD[1...5]: gespeicherte Fehler, der jüngste Fehler steht im Index [1]

CANx_SLAVE_NODEID

2044

= CANx Slave Node-ID

x = 1...n = Nummer der CAN-Schnittstelle (je nach Gerät, → Datenblatt)

Baustein-Typ = Funktionsbaustein (FB)

Baustein ist enthalten in Bibliothek `ifm_CR7132_CANopenxSlave_Vxxyyzz.LIB`

Symbol in CODESYS:



Beschreibung

2049

CANx_SLAVE_NODEID ermöglicht das Einstellen der Node-ID eines CANopen-Slaves zur Laufzeit des Anwendungsprogramms.

Der FB wird im Normalfall bei der Initialisierung der Steuerung einmalig, im ersten Zyklus, aufgerufen. Anschließend wird der Eingang ENABLE wieder auf FALSE gesetzt.

Parameter der Eingänge

2047

Parameter	Datentyp	Beschreibung
ENABLE	BOOL	FALSE ⇨ TRUE (Flanke): Parameter übernehmen und aktivieren sonst: diese Funktion wird nicht ausgeführt
NODEID	BYTE	Node-ID = ID des Knotens zulässige Werte = 0...127

CANx_SLAVE_SEND_EMERGENCY

2056

x = 1...n = Nummer der CAN-Schnittstelle (je nach Gerät, → Datenblatt)

Baustein-Typ = Funktionsbaustein (FB)

Baustein ist enthalten in Bibliothek `ifm_CR7132_CANopenxSlave_Vxxyyzz.LIB`

Symbol in CODESYS:

CANx_SLAVE_SEND_EMERGENCY
- ENABLE
- ERROR
- ERROR_CODE
- ERROR_REGISTER
- MANUFACTURER_ERROR_FIELD

Beschreibung

2059

CANx_SLAVE_SEND_EMERGENCY versendet anwendungsspezifische Fehlerstatus. Das sind Fehlernachrichten, die zusätzlich zu den geräteinternen Fehlernachrichten (z.B. Kurzschluss am Ausgang) gesendet werden sollen.

► Den FB aufrufen, wenn der Fehlerstatus an andere Geräte im Netzwerkverbund übertragen werden soll.

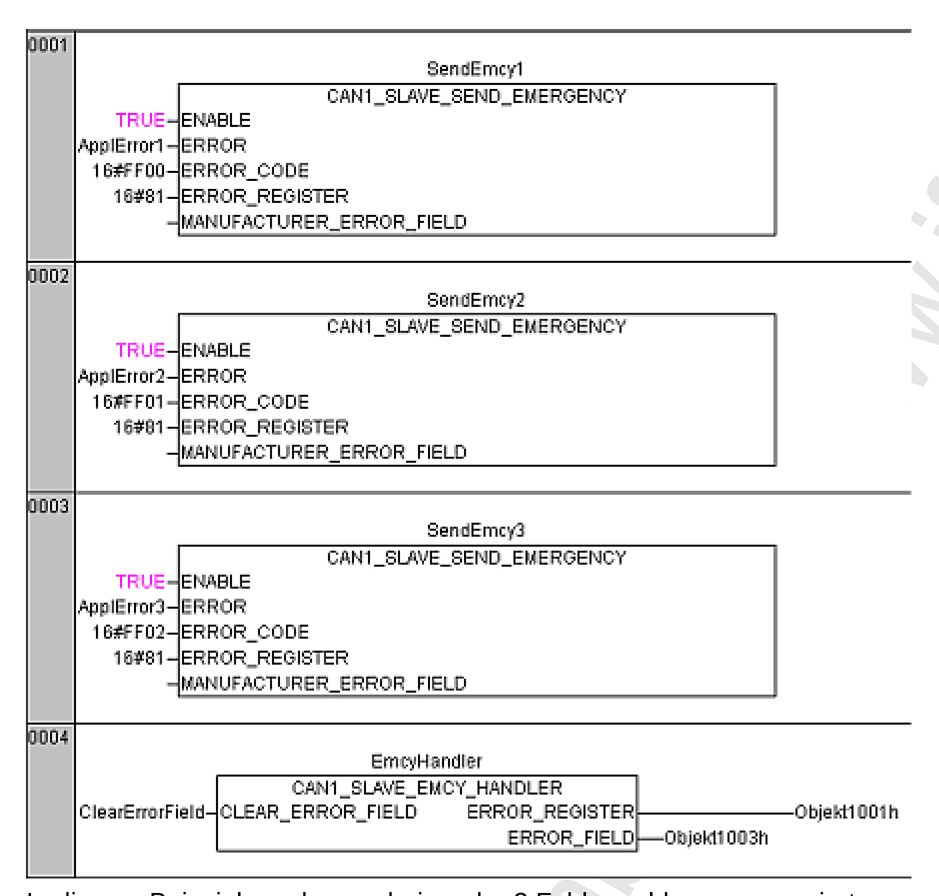
Parameter der Eingänge

2060

Parameter	Datentyp	Beschreibung
ENABLE	BOOL	TRUE: Baustein ausführen FALSE: Baustein wird nicht ausgeführt > Baustein-Eingänge sind nicht aktiv > Baustein-Ausgänge sind nicht spezifiziert
ERROR	BOOL	FALSE ⇨ TRUE (Flanke): sendet den anstehenden Fehler-Code TRUE ⇨ FALSE (Flanke) UND Fehler steht nicht mehr an: Nach Verzögerung von ca. 1 s: > Null-Fehlermeldung wird gesendet sonst: diese Funktion wird nicht ausgeführt
ERROR_CODE	WORD	Der Error-Code gibt detailliert Auskunft über den erkannten Fehler. Die Werte sollten gemäß der CANopen-Spezifikation eingetragen werden. → Kapitel ***Übersicht CANopen Error-Codes*** (→ Seite 415)
ERROR_REGISTER	BYTE	ERROR_REGISTER gibt die Art des Fehlers an. Der hier angegebene Wert wird mit allen anderen aktuell aktiven Fehlernachrichten bitweise ODER-verknüpft. Der sich hierbei ergebende Wert wird ins Error-Register (Index 1001_{16}/00) geschrieben und mit der EMCY-Nachricht versendet. Die Werte sollten gemäß der CANopen-Spezifikation eingetragen werden.
MANUFACTURER_ERROR_FIELD	ARRAY [0..4] OF BYTE	Hier können bis zu 5 Bytes anwendungsspezifische Fehlerinformationen eingetragen werden. Das Format ist dabei frei wählbar.

Beispiel: CANx_SLAVE_SEND_EMERGENCY

2062



In diesem Beispiel werden nacheinander 3 Fehlermeldungen generiert:

1. ApplError1, Code = 0xFF00 im Fehlerregister 0x81
2. ApplError2, Code = 0xFF01 im Fehlerregister 0x81
3. ApplError3, Code = 0xFF02 im Fehlerregister 0x81

Der FB CAN1_SLAVE_EMCY_HANDLER sendet die Fehlermeldungen an das Fehler-Register "Objekt 0x1001" im Fehler-Array "Objekt 0x1003".

CANx_SLAVE_SET_PREOP

2700

x = 1...n = Nummer der CAN-Schnittstelle (je nach Gerät, → Datenblatt)

Baustein-Typ = Funktionsbaustein (FB)

Baustein ist enthalten in Bibliothek `ifm_CR7132_CANopenxSlave_Vxxyyzz.LIB`

Symbol in CODESYS:



Beschreibung

2703

CANx_SLAVE_SET_PREOP schaltet den Betriebsmodus dieses CANopen-Slaves von OPERATIONAL auf PRE-OPERATIONAL.

Normalerweise schaltet das Gerät im Fehlerfall lediglich die Ausgänge ab. Unter bestimmten Bedingungen kann es erforderlich sein, dass das Anwendungsprogramm den Betriebszustand des als Slave arbeitenden Geräts auf PRE-OPERATIONAL setzt. Dies erfolgt über den hier beschriebenen FB.

Parameter der Eingänge

2704

Parameter	Datentyp	Beschreibung
ENABLE	BOOL	FALSE ⇨ TRUE (Flanke): Slave auf PRE-OPERATIONAL setzen sonst: diese Funktion wird nicht ausgeführt

CANx_SLAVE_STATUS

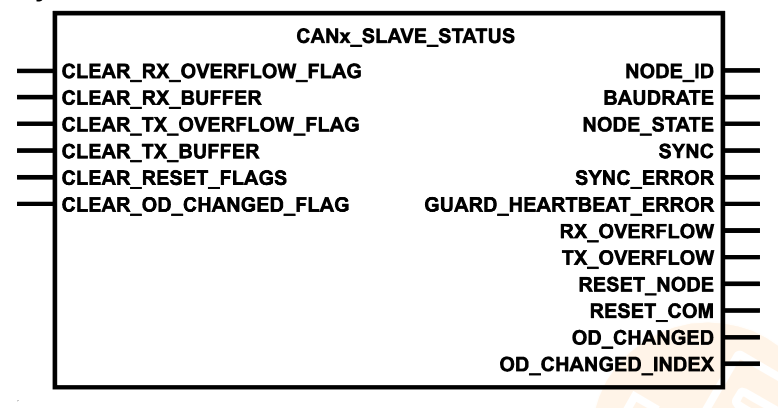
2706

x = 1...n = Nummer der CAN-Schnittstelle (je nach Gerät, → Datenblatt)

Baustein-Typ = Funktionsbaustein (FB)

Baustein ist enthalten in Bibliothek `ifm_CR7132_CANopenxSlave_Vxxyyzz.LIB`

Symbol in CODESYS:



Beschreibung

2707

CANx_SLAVE_STATUS zeigt den Status des als CANopen-Slave eingesetzten Gerätes.

! Wir empfehlen dringend, die Auswertung des Netzwerkstatus über diesen FB vorzunehmen.

Parameter der Eingänge

2708

Parameter	Datentyp	Beschreibung
CLEAR_RX_OVERFLOW_FLAG	BOOL	FALSE ⇨ TRUE (Flanke): Fehlerflag RX_OVERFLOW löschen sonst: diese Funktion wird nicht ausgeführt
CLEAR_RX_BUFFER	BOOL	FALSE ⇨ TRUE (Flanke): Daten im Empfangspuffer löschen sonst: diese Funktion wird nicht ausgeführt
CLEAR_TX_OVERFLOW_FLAG	BOOL	FALSE ⇨ TRUE (Flanke): Fehlerflag TX_OVERFLOW löschen sonst: diese Funktion wird nicht ausgeführt
CLEAR_TX_BUFFER	BOOL	FALSE ⇨ TRUE (Flanke): Daten im Sendepuffer löschen sonst: diese Funktion wird nicht ausgeführt
CLEAR_RESET_FLAGS	BOOL	FALSE ⇨ TRUE (Flanke): Flag RESET_NODE löschen Flag RESET_COM löschen sonst: diese Funktion wird nicht ausgeführt
CLEAR_OD_CHANGED_FLAGS	BOOL	FALSE ⇨ TRUE (Flanke): Flag OD_CHANGED löschen Flag OD_CHANGED_INDEX löschen sonst: diese Funktion wird nicht ausgeführt

Parameter der Ausgänge

2068

Parameter	Datentyp	Beschreibung
NODE_ID	BYTE	aktueller Knoten-ID
BAUDRATE	WORD	aktuelle Baudrate des Knotens in [kBaud]
NODE_STATE	BYTE	aktueller Status des CANopen-Slaves 0 = Bootup-Nachricht versendet 4 = CANopen-Slave im Status PRE-OPERATIONAL und wird per SDO-Zugriff konfiguriert 5 = CANopen-Slave im Status OPERATIONAL 127 = CANopen-Slave im Status PRE-OPERATIONAL
SYNC	BOOL	SYNC-Signal des CANopen-Masters TRUE: Im letzten Zyklus wurde ein SYNC-Signal empfangen FALSE: Im letzten Zyklus wurde kein SYNC-Signal empfangen
SYNC_ERROR	BOOL	Fehler: das SYNC-Signal des Masters wurde nicht oder zu spät empfangen
GUARD_HEARTBEAT_ERROR	BOOL	Fehler: das Guard- oder Heartbeat-Signal des Masters wurde nicht oder zu spät empfangen
RX_OVERFLOW	BOOL	Fehler: Empfangspuffer-Überlauf
TX_OVERFLOW	BOOL	Fehler: Sendepuffer-Überlauf
RESET_NODE	BOOL	Der CANopen-Stack des Slaves wurde vom Master zurückgesetzt.
RESET_COM	BOOL	Das Kommunikations-Interface des CAN-Stack wurde vom Master zurückgesetzt.
OD_CHANGED	BOOL	Daten im Objektverzeichnis des CANopen-Masters wurden geändert
OD_CHANGED_INDEX	INT	Index des zuletzt geänderten Objektverzeichnis-Eintrags

6.2.6 Bausteine: CANopen SDOs

Inhalt

2071

Hier finden Sie **ifm**-Bausteine für den Umgang von CANopen mit Service Data Objects (SDOs).

CANx_SDO_READ

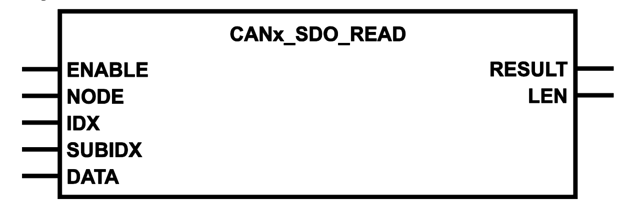
621

x = 1...n = Nummer der CAN-Schnittstelle (je nach Gerät, → Datenblatt)

Baustein-Typ = Funktionsbaustein (FB)

Baustein ist enthalten in Bibliothek `ifm_CR7132_Vxxyyzz.LIB`

Symbol in CODESYS:



Beschreibung

624

CANx_SDO_READ liest das →***SDO*** (→ Seite 484) mit den angegebenen Indizes aus dem Knoten aus.

Voraussetzung: Knoten muss sich im Zustand PRE-OPERATIONAL oder OPERATIONAL befinden.

Über diese Indizes können die Einträge im Objektverzeichnis gelesen werden. Dadurch ist es möglich, die Knotenparameter gezielt zu lesen.

Beispiel:

0004
SDO_read1
sdo1_data – ADR
m1 – ENABLE, node – NODE, idx – IDX, subidx – SUBIDX, DATA
CAN1_SDO_READ
RESULT – EQ – R – m1
LEN – LEN1 – EQ

Parameter der Eingänge

625

Parameter	Datentyp	Beschreibung
ENABLE	BOOL	TRUE: Baustein ausführen FALSE: Baustein wird nicht ausgeführt > Baustein-Eingänge sind nicht aktiv > Baustein-Ausgänge sind nicht spezifiziert
NODE	BYTE	CANopen-ID des Knotens zulässig = 1...127 = 0x01...0x7F
IDX	WORD	Index im Objektverzeichnis
SUBIDX	BYTE	Subindex bezogen auf den Index im Objektverzeichnis
DATA	DWORD	Adresse des Empfangsdaten-Arrays zulässige Länge = 0...255 (!) Die Adresse mit dem Operator ADR ermitteln und dem FB übergeben!

Parameter der Ausgänge

626

Parameter	Datentyp	Beschreibung
RESULT	BYTE	Rückmeldung des Funktionsbausteins (mögliche Meldungen → folgende Tabelle)
LEN	WORD	Länge des Eintrags in "Anzahl der Bytes" Der Wert für LEN darf nicht größer sein als die Größe des Empfangs-Arrays. Andernfalls werden beliebige Daten in der Anwendung überschrieben.

Mögliche Ergebnisse für RESULT:

Wert dez	hex	Beschreibung
0	00	FB ist inaktiv
1	01	FB-Ausführung wurde ohne Fehler beendet – Daten sind gültig
2	02	Funktionsbaustein ist aktiv (Aktion noch nicht beendet)
3	03	Fehler, keine Daten während der Überwachungszeit empfangen

CANx_SDO_WRITE

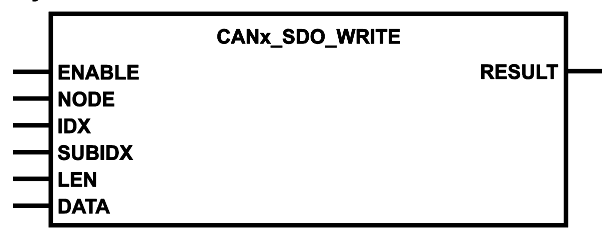
615

x = 1...n = Nummer der CAN-Schnittstelle (je nach Gerät, → Datenblatt)

Baustein-Typ = Funktionsbaustein (FB)

Baustein ist enthalten in Bibliothek `ifm_CR7132_Vxxyyzz.LIB`

Symbol in CODESYS:



Beschreibung

618

CANx_SDO_WRITE schreibt das →***SDO*** (→ Seite 484) mit den angegebenen Indizes in den Knoten.

Voraussetzung: Knoten muss sich im Zustand PRE-OPERATIONAL oder OPERATIONAL befinden.

Über diesen FB können die Einträge im Objektverzeichnis geschrieben werden. Dadurch ist es möglich, die Knotenparameter gezielt zu setzen.

> ❗ Der Wert für LEN muss kleiner sein als die Größe des Sende-Arrays. Andernfalls werden beliebige Daten versendet.

Beispiel:

0005
SDO_write1
sdo1_data ADR
CAN1_SDO_WRITE
m1 ENABLE RESULT
node NODE
idx IDX
subidx SUBIDX
sdo_len LEN
DATA
1 EQ
R
m1

Parameter der Eingänge

619

Parameter	Datentyp	Beschreibung
ENABLE	BOOL	TRUE: Baustein ausführen FALSE: Baustein wird nicht ausgeführt > Baustein-Eingänge sind nicht aktiv > Baustein-Ausgänge sind nicht spezifiziert
NODE	BYTE	CANopen-ID des Knotens zulässig = 1...127 = 0x01...0x7F
IDX	WORD	Index im Objektverzeichnis
SUBIDX	BYTE	Subindex bezogen auf den Index im Objektverzeichnis
LEN	WORD	Länge des Eintrags in "Anzahl der Bytes" Der Wert für LEN darf nicht größer sein als die Größe des Sende-Arrays. Andernfalls werden beliebige Daten versendet.
DATA	DWORD	Adresse des Sendedaten-Arrays zulässige Länge = 0...255 Die Adresse mit dem Operator ADR ermitteln und dem FB übergeben!

Parameter der Ausgänge

620

Parameter	Datentyp	Beschreibung
RESULT	BYTE	Rückmeldung des Funktionsbausteins (mögliche Meldungen → folgende Tabelle)

Mögliche Ergebnisse für RESULT:

Wert dez	Wert hex	Beschreibung
0	00	FB ist inaktiv
1	01	FB-Ausführung wurde ohne Fehler beendet – Daten sind gültig
2	02	Funktionsbaustein ist aktiv (Aktion noch nicht beendet)
3	03	Fehler, Daten können nicht übertragen werden

6.2.7 Bausteine: SAE J1939

Inhalt

2273

Für SAE J1939 stellt **ifm electronic** eine Reihe von Bausteinen zur Verfügung, die im Folgenden erklärt werden.

J1939_x

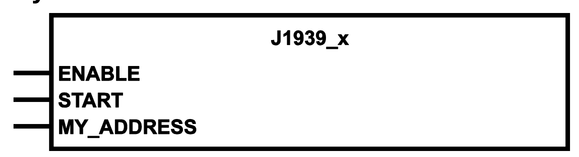
2274

x = 1...n = Nummer der CAN-Schnittstelle (je nach Gerät, → Datenblatt)

Baustein-Typ = Funktionsbaustein (FB)

Baustein ist enthalten in Bibliothek `ifm_CR7132_J1939_Vxxyyzz.LIB`

Symbol in CODESYS:



Beschreibung

2276

J1939_x dient als Protokoll-Handler für das Kommunikationsprofil SAE J1939.

Zur Abwicklung der Kommunikation muss der Protokoll-Handler in jedem Programmzyklus aufgerufen werden. Dazu wird der Eingang ENABLE auf TRUE gesetzt.

Der Protokoll-Handler wird gestartet, wenn der Eingang START für einen Zyklus auf TRUE gesetzt wird.

Über MY_ADRESS wird dem Controller eine Geräteadresse übergeben. Sie muss sich von Adressen der anderen J1939-Busteilnehmer unterscheiden. Sie kann dann von anderen Busteilnehmern ausgelesen werden.

Parameter der Eingänge

469

Parameter	Datentyp	Beschreibung
ENABLE	BOOL	TRUE: Baustein ausführen FALSE: Baustein wird nicht ausgeführt > Baustein-Eingänge sind nicht aktiv > Baustein-Ausgänge sind nicht spezifiziert
START	BOOL	TRUE (nur 1 Zyklus lang): J1939-Protokoll an CAN-Schnittstelle x starten FALSE: im weiteren Programmablauf
MY_ADDRESS	BYTE	J1939-Adresse des Geräts

J1939_x_GLOBAL_REQUEST

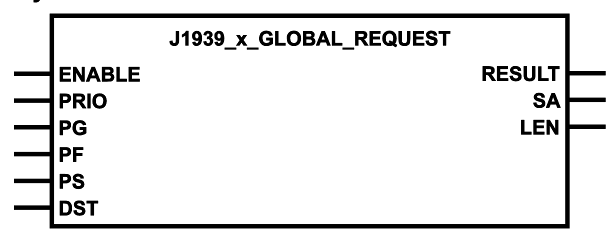
2282

x = 1...n = Nummer der CAN-Schnittstelle (je nach Gerät, → Datenblatt)

Baustein-Typ = Funktionsbaustein (FB)

Baustein ist enthalten in Bibliothek `ifm_CR7132_J1939_Vxxyyzz.LIB`

Symbol in CODESYS:



Beschreibung

2301

J1939_x_GLOBAL_REQUEST ist für das automatische Anfordern einzelner Nachrichten von allen (global) aktiven J1939-Netzwerkteilnehmern verantwortlich. Dazu werden dem FB die Parameter PG, PF, PS und die Adresse des Arrays DST übergeben, in dem die empfangenen Daten abgelegt werden.

Info

PGN = [Page] + [PF] + [PS]
PDU = [PRIO] + [PGN] + [J1939-Adresse] + [Daten]

13790

ACHTUNG

Daten können unzulässig überschrieben werden!

- ► Ein Empfangs-Array mit einer Größe von 1 785 Bytes anlegen!
  Dies ist die maximale Größe einer J1939-Nachricht.
- ► Die Anzahl empfangener Daten prüfen:
  der Wert darf nicht größer sein als das bereitgestellte Empfangs-Array!

- ► Für jede angefragte Nachricht eine eigene Instanz des FBs verwenden!
- ► Für die Zieladresse DST gilt:
  Die Adresse mit dem Operator ADR ermitteln und dem FB übergeben!
- ► Zusätzlich die Priorität (typisch 3, 6 oder 7) übergeben.
- ► Da das Anfordern der Daten über mehrere Steuerungszyklen abgewickelt werden kann, muss dieser Vorgang über das RESULT-Byte ausgewertet werden. Wird RESULT = 1, wurden alle Daten empfangen.

> Der Ausgang LEN zeigt an, wie viele Datenbytes empfangen wurden.

> Der Empfang einer neuen Nachricht überschreibt die Daten auf der Speicheradresse DST.

> Wird innerhalb von 1,25 Sekunden von keinem Teilnehmer am Bus eine Antwort gesendet, geht der FB wieder in den inaktiven Zustand (⇨ RESULT = 0).

Parameter der Eingänge

463

Parameter	Datentyp	Beschreibung
ENABLE	BOOL	TRUE: Baustein ausführen FALSE: Baustein wird nicht ausgeführt > Baustein-Eingänge sind nicht aktiv > Baustein-Ausgänge sind nicht spezifiziert
PRIO	BYTE	Nachrichten-Prioritätin der PDU (Parameter Data Unit) zulässig = 0…7
PG	BYTE	Data Page Wert der definierten PGN (Parameter Group Number) zulässig = 0...1 (normalerweise = 0)
PF	BYTE	PDU format byte Wert der definierten PGN (Parameter Group Number) PDU2 (global) = 240...254
PS	BYTE	PDU specific byte Wert der definierten PGN (Parameter Group Number) GE (Group Extension) = 0...255
DST	DWORD	Startadresse im Zielspeicher ! Die Adresse mit dem Operator ADR ermitteln und dem FB übergeben!

Info

PGN = [Page] + [PF] + [PS]
PDU = [PRIO] + [PGN] + [J1939-Adresse] + [Daten]

Parameter der Ausgänge

464

Parameter	Datentyp	Beschreibung
RESULT	BYTE	Rückmeldung des Funktionsbausteins (mögliche Meldungen → folgende Tabelle)
SA	BYTE	J1939-Adresse des antwortenden Geräts
LEN	WORD	Anzahl der empfangenen Bytes

Mögliche Ergebnisse für RESULT:

Wert dez	Wert hex	Beschreibung
0	00	FB ist inaktiv
1	01	FB-Ausführung wurde ohne Fehler beendet – Daten sind gültig
2	02	Funktionsbaustein ist aktiv (Aktion noch nicht beendet)

J1939_x_RECEIVE

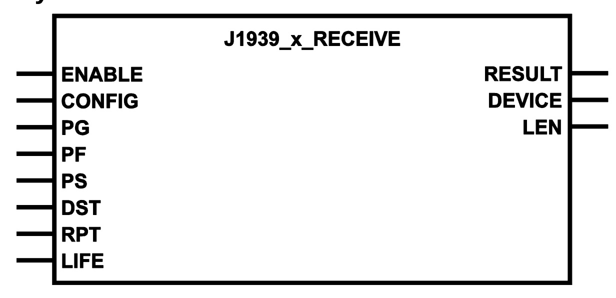
2278

x = 1...n = Nummer der CAN-Schnittstelle (je nach Gerät, → Datenblatt)

Baustein-Typ = Funktionsbaustein (FB)

Baustein ist enthalten in Bibliothek `ifm_CR7132_J1939_Vxxyyzz.LIB`

Symbol in CODESYS:



Beschreibung

2288

J1939_x_RECEIVE dient dem Empfang einer einzelnen Nachricht oder eines Nachrichtenblocks.

Dazu muss der FB über den Eingang CONFIG für einen Zyklus initialisiert werden. Bei der Initialisierung werden die Parameter PG, PF, PS, RPT, LIFE und die Speicheradresse des Datenarrays DST übergeben.

> (!) Nach dem ersten Konfigurieren können diese Parameter im laufenden Anwendungsprogramm nicht mehr verändert werden: PG, PF, PS, RPT, LIFE, DST.

13790

ACHTUNG

Daten können unzulässig überschrieben werden!

- ► Ein Empfangs-Array mit einer Größe von 1 785 Bytes anlegen!
  Dies ist die maximale Größe einer J1939-Nachricht.
- ► Die Anzahl empfangener Daten prüfen:
  der Wert darf nicht größer sein als das bereitgestellte Empfangs-Array!

- ► Für die Zieladresse DST gilt:
  (!) Die Adresse mit dem Operator ADR ermitteln und dem FB übergeben!

(!) Nach dem ersten Setzen kann RPT nicht mehr verändert werden!

- ► Der Datenempfang muss über das RESULT-Byte ausgewertet werden. Wird RESULT = 1, können die Daten von der über DST übergebenen Speicheradresse ausgelesen und weiter verarbeitet werden.
- > Der Empfang einer neuen Nachricht überschreibt die Daten auf der Speicheradresse DST.
- > Die Anzahl der empfangenen Nachrichten-Bytes wird über den Ausgang LEN angezeigt.
- > Wird RESULT = 3, wurden im angegebenen Zeitfenster (LIFE • RPT) keine gültigen Nachrichten empfangen.

> (!) Dieser Baustein muss auch eingesetzt werden, wenn die Nachrichten mit den FBs J1939_..._REQUEST angefordert werden.

Parameter der Eingänge

457

Parameter	Datentyp	Beschreibung
ENABLE	BOOL	TRUE: Baustein ausführen FALSE: Baustein wird nicht ausgeführt > Baustein-Eingänge sind nicht aktiv > Baustein-Ausgänge sind nicht spezifiziert
CONFIG	BOOL	TRUE (im 1. Zyklus): Datenobjekt konfigurieren FALSE: im weiteren Programmablauf
PG	BYTE	Data Page Wert der definierten PGN (Parameter Group Number) zulässig = 0...1 (normalerweise = 0)
PF	BYTE	PDU format byte Wert der definierten PGN (Parameter Group Number) PDU1 (specific) = 0...239 PDU2 (global) = 240...255
PS	BYTE	PDU specific byte Wert der definierten PGN (Parameter Group Number) Wenn PF = PDU1 ⇨ PS = DA (Destination Address) (DA = J1939-Adresse des externen Geräts) Wenn PF = PDU2 ⇨ PS = GE (Group Extension)
DST	DWORD	Startadresse im Zielspeicher Die Adresse mit dem Operator ADR ermitteln und dem FB übergeben!
RPT	TIME	Überwachungszeit Innerhalb dieses angegebenen Zeitfensters müssen die Telegramme zyklisch empfangen werden. > Andernfalls erfolgt eine Fehlermeldung. RPT = T#0s ⇨ keine Überwachung Nach dem ersten Setzen kann RPT nicht mehr verändert werden!
LIFE	BYTE	tolerierte Anzahl der nicht empfangenen J1939-Nachrichten

Parameter der Ausgänge

458

Parameter	Datentyp	Beschreibung
RESULT	BYTE	Rückmeldung des Funktionsbausteins (mögliche Meldungen → folgende Tabelle)
DEVICE	BYTE	J1939-Adresse des Absenders
LEN	WORD	Anzahl der empfangenen Bytes

Mögliche Ergebnisse für RESULT:

Wert dez	Wert hex	Beschreibung
0	00	FB ist inaktiv
1	01	FB-Ausführung wurde ohne Fehler beendet – Daten sind gültig
3	03	Fehler, keine Daten während der Überwachungszeit empfangen

J1939_x_RESPONSE

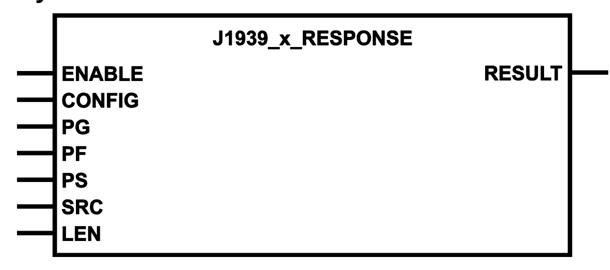
2280

x = 1...n = Nummer der CAN-Schnittstelle (je nach Gerät, → Datenblatt)

Baustein-Typ = Funktionsbaustein (FB)

Baustein ist enthalten in Bibliothek `ifm_CR7132_J1939_Vxxyyzz.LIB`

Symbol in CODESYS:



Beschreibung

2299

J1939_x_RESPONSE organisiert die automatische Antwort auf ein Request-Telegramm (Anforderungstelegramm).

Der FB ist für das automatische Versenden von Nachrichten auf "Global Requests" und "Specific Requests" verantwortlich. Dazu muss der FB über den Eingang CONFIG für einen Zyklus initialisiert werden.

Dem FB werden die Parameter PG, PF, PS, RPT und die Adresse des Datenarrays SRC übergeben.

► Für die Quelladresse SRC gilt:
  (!) Die Adresse mit dem Operator ADR ermitteln und dem FB übergeben!

► Zusätzlich die Anzahl der zu übertragenen Datenbytes übergeben.

Parameter der Eingänge

451

Parameter	Datentyp	Beschreibung
ENABLE	BOOL	TRUE: Baustein ausführen FALSE: Baustein wird nicht ausgeführt > Baustein-Eingänge sind nicht aktiv > Baustein-Ausgänge sind nicht spezifiziert
CONFIG	BOOL	TRUE (im 1. Zyklus): Datenobjekt konfigurieren FALSE: im weiteren Programmablauf
PG	BYTE	Data Page Wert der definierten PGN (Parameter Group Number) zulässig = 0...1 (normalerweise = 0)
PF	BYTE	PDU format byte Wert der definierten PGN (Parameter Group Number) PDU1 (specific) = 0...239 PDU2 (global) = 240...255
PS	BYTE	PDU specific byte Wert der definierten PGN (Parameter Group Number) Wenn PF = PDU1 ⇨ PS = DA (Destination Address) (DA = J1939-Adresse des externen Geräts) Wenn PF = PDU2 ⇨ PS = GE (Group Extension)
SRC	DWORD	Startadresse im Quellspeicher ! Die Adresse mit dem Operator ADR ermitteln und dem FB übergeben!
LEN	WORD	Anzahl (≥ 1) der zu übertragenden Daten-Bytes

Parameter der Ausgänge

13993

Parameter	Datentyp	Beschreibung
RESULT	BYTE	Rückmeldung des Funktionsbausteins (mögliche Meldungen → folgende Tabelle)

Mögliche Ergebnisse für RESULT:

Wert dez	Wert hex	Beschreibung
0	00	FB ist inaktiv
1	01	Datenübertragung wurde ohne Fehler beendet
2	02	Funktionsbaustein ist aktiv (Aktion noch nicht beendet)
3	03	Fehler, Daten können nicht übertragen werden

J1939_x_SPECIFIC_REQUEST

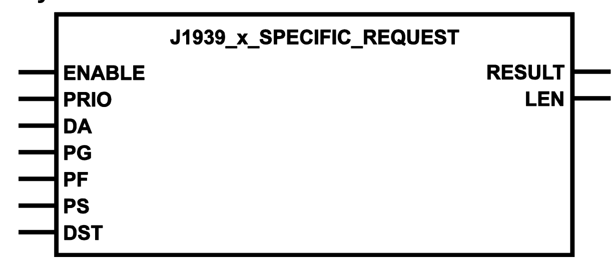
2281

x = 1...n = Nummer der CAN-Schnittstelle (je nach Gerät, → Datenblatt)

Baustein-Typ = Funktionsbaustein (FB)

Baustein ist enthalten in Bibliothek `ifm_CR7132_J1939_Vxxyyzz.LIB`

Symbol in CODESYS:



Beschreibung

2300

J1939_x_SPECIFIC_REQUEST ist für das automatische Anfordern einzelner Nachrichten von einem bestimmten (specific) J1939-Netzwerkteilnehmer verantwortlich. Dazu werden dem FB die logische Geräteadresse DA, die Parameter PG, PF, PS und die Adresse des Arrays DST übergeben, in dem die empfangenen Daten abgelegt werden.

Info

PGN = [Page] + [PF] + [PS]
PDU = [PRIO] + [PGN] + [J1939-Adresse] + [Daten]

13790

ACHTUNG

Daten können unzulässig überschrieben werden!

► Ein Empfangs-Array mit einer Größe von 1 785 Bytes anlegen!
Dies ist die maximale Größe einer J1939-Nachricht.

► Die Anzahl empfangener Daten prüfen:
der Wert darf nicht größer sein als das bereitgestellte Empfangs-Array!

► Für die Zieladresse gilt:
Die Adresse mit dem Operator ADR ermitteln und dem FB übergeben!

► Zusätzlich die Priorität (typisch 3, 6 oder 7) übergeben.

► Da das Anfordern der Daten über mehrere Steuerungszyklen abgewickelt werden kann, muss dieser Vorgang über das RESULT-Byte ausgewertet werden. Wird RESULT = 1, wurden alle Daten empfangen.

> Der Ausgang LEN zeigt an, wie viele Datenbytes empfangen wurden.

> Wird innerhalb von 1,25 Sekunden vom angeforderten Teilnehmer keine Antwort gesendet, meldet der FB einen Fehler (⇨ RESULT = 3).

Parameter der Eingänge

445

Parameter	Datentyp	Beschreibung
ENABLE	BOOL	TRUE: Baustein ausführen FALSE: Baustein wird nicht ausgeführt > Baustein-Eingänge sind nicht aktiv > Baustein-Ausgänge sind nicht spezifiziert
PRIO	BYTE	Nachrichten-Prioritätin der PDU (Parameter Data Unit) zulässig = 0…7
DA	BYTE	J1939-Adresse des angefragten Geräts
PG	BYTE	Data Page Wert der definierten PGN (Parameter Group Number) zulässig = 0...1 (normalerweise = 0)
PF	BYTE	PDU format byte Wert der definierten PGN (Parameter Group Number) PDU1 (specific) = 0...239 PDU2 (global) = 240...255
PS	BYTE	PDU specific byte Wert der definierten PGN (Parameter Group Number) Wenn PF = PDU1 ⇨ PS = DA (Destination Address) (DA = J1939-Adresse des externen Geräts) Wenn PF = PDU2 ⇨ PS = GE (Group Extension)
DST	DWORD	Startadresse im Zielspeicher ❗ Die Adresse mit dem Operator ADR ermitteln und dem FB übergeben!

Info

PGN = [Page] + [PF] + [PS]
PDU = [PRIO] + [PGN] + [J1939-Adresse] + [Daten]

Parameter der Ausgänge

446

Parameter	Datentyp	Beschreibung
RESULT	BYTE	Rückmeldung des Funktionsbausteins (mögliche Meldungen → folgende Tabelle)
LEN	WORD	Anzahl der empfangenen Bytes

Mögliche Ergebnisse für RESULT:

Wert dez	hex	Beschreibung
0	00	FB ist inaktiv
1	01	FB-Ausführung wurde ohne Fehler beendet – Daten sind gültig
2	02	Funktionsbaustein ist aktiv (Aktion noch nicht beendet)
3	03	Fehler

J1939_x_TRANSMIT

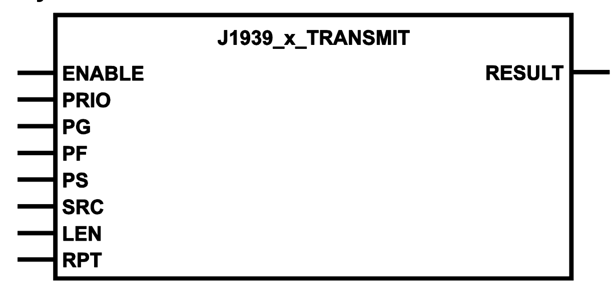
279

x = 1...n = Nummer der CAN-Schnittstelle (je nach Gerät, → Datenblatt)

Baustein-Typ = Funktionsbaustein (FB)

Baustein ist enthalten in Bibliothek `ifm_CR7132_J1939_Vxxyyzz.LIB`

Symbol in CODESYS:



Beschreibung

2298

J1939_x_TRANSMIT ist für das Versenden einzelner Nachrichten oder Nachrichtenblocks verantwortlich. Dazu werden dem FB die Parameter PG, PF, PS, RPT und die Adresse des Datenarrays SRC übergeben.

Info

PGN = [Page] + [PF] + [PS]
PDU = [PRIO] + [PGN] + [J1939-Adresse] + [Daten]

- ► Für die Quelladresse SRC gilt:
  Die Adresse mit dem Operator ADR ermitteln und dem FB übergeben!
- ► Zusätzlich die Anzahl der zu übertragenen Datenbytes und die Priorität (typisch 3, 6 oder 7) übergeben.
- ► Da das Versenden der Daten über mehrere Steuerungszyklen abgewickelt wird, muss der Vorgang über das RESULT-Byte ausgewertet werden. Wird RESULT = 1, wurden alle Daten übertragen.

Wenn mehr als 8 Bytes gesendet werden sollen, wird ein "multi package transfer" durchgeführt.

Parameter der Eingänge

439

Parameter	Datentyp	Beschreibung
ENABLE	BOOL	TRUE: Baustein ausführen FALSE: Baustein wird nicht ausgeführt > Baustein-Eingänge sind nicht aktiv > Baustein-Ausgänge sind nicht spezifiziert
PRIO	BYTE	Nachrichten-Prioritätin der PDU (Parameter Data Unit) zulässig = 0…7
PG	BYTE	Data Page Wert der definierten PGN (Parameter Group Number) zulässig = 0...1 (normalerweise = 0)
PF	BYTE	PDU format byte Wert der definierten PGN (Parameter Group Number) PDU1 (specific) = 0...239 PDU2 (global) = 240...255
PS	BYTE	PDU specific byte Wert der definierten PGN (Parameter Group Number) Wenn PF = PDU1 ⇨ PS = DA (Destination Address) (DA = J1939-Adresse des externen Geräts) Wenn PF = PDU2 ⇨ PS = GE (Group Extension)
SRC	DWORD	Startadresse im Quellspeicher Die Adresse mit dem Operator ADR ermitteln und dem FB übergeben!
LEN	WORD	Anzahl der zu übertragenden Daten-Bytes zulässig = 1...1 785 = 0x0001...0x06F9
RPT	TIME	Wiederholzeit, innerhalb der die Daten-Telegramme zyklisch versendet werden sollen RPT = T#0s ⇨ nur einmalig versenden

Info

PGN = [Page] + [PF] + [PS]
PDU = [PRIO] + [PGN] + [J1939-Adresse] + [Daten]

Parameter der Ausgänge

440

Parameter	Datentyp	Beschreibung
RESULT	BYTE	Rückmeldung des Funktionsbausteins (mögliche Meldungen → folgende Tabelle)

Mögliche Ergebnisse für RESULT:

Wert dez	Wert hex	Beschreibung
0	00	FB ist inaktiv
1	01	FB-Ausführung wurde ohne Fehler beendet – Daten sind gültig
2	02	Funktionsbaustein ist aktiv (Aktion noch nicht beendet)
3	03	Fehler, Daten können nicht übertragen werden

6.2.8 Bausteine: serielle Schnittstelle

Inhalt

13011

HINWEIS

Voreingestellt steht die serielle Schnittstelle dem Anwender nicht zur Verfügung, da sie für den Programm-Download und das Debugging genutzt wird.

Setzt der Anwender das Systemmerkerbit SERIAL_MODE=TRUE, dann kann die Schnittstelle frei genutzt werden. Ein Debugging des Anwendungsprogramms ist dann nur noch über eine der 4 CAN-Schnittstellen oder über USB möglich.

Mit den folgend aufgeführten Bausteinen kann die serielle Schnittstelle im Anwendungsprogramm genutzt werden.

SERIAL_PENDING

314

Baustein-Typ = Funktionsbaustein (FB)

Baustein ist enthalten in Bibliothek `ifm_CR7132_Vxxyyzz.LIB`

Symbol in CODESYS:



Beschreibung

12994

SERIAL_PENDING ermittelt die Anzahl der im seriellen Empfangspuffer gespeicherten Datenbytes.

Im Gegensatz zu ***SERIAL_RX*** (→ Seite 257) bleibt der Inhalt des Puffers nach Aufruf dieser Funktion unverändert.

Die SERIAL-Bausteine bilden die Grundlage für die Erstellung eines anwendungsspezifischen Protokolls für die serielle Schnittstelle.

> **HINWEIS**
>
> Voreingestellt steht die serielle Schnittstelle dem Anwender nicht zur Verfügung, da sie für den Programm-Download und das Debugging genutzt wird.
>
> Setzt der Anwender das Systemmerkerbit SERIAL_MODE=TRUE, dann kann die Schnittstelle frei genutzt werden. Ein Debugging des Anwendungsprogramms ist dann nur noch über eine der 4 CAN-Schnittstellen oder über USB möglich.

Parameter der Ausgänge

12996

Parameter	Datentyp	Beschreibung
NUMBER	WORD	Anzahl der empfangenen Datenbytes (1...1 000)

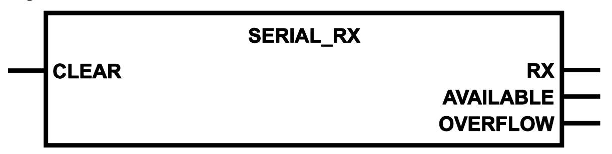
SERIAL_RX

308

Baustein-Typ = Funktionsbaustein (FB)

Baustein ist enthalten in Bibliothek `ifm_CR7132_Vxxyyzz.LIB`

Symbol in CODESYS:



Beschreibung

12997

SERIAL_RX liest mit jedem Aufruf ein empfangenes Datenbyte aus dem seriellen Empfangspuffer aus.

Gehen mehr als 1 000 Datenbytes ein, läuft der Puffer über und es gehen Daten verloren. Dieses wird durch das Bit OVERFLOW angezeigt.

Die SERIAL-Bausteine bilden die Grundlage für die Erstellung eines anwendungsspezifischen Protokolls für die serielle Schnittstelle.

> **HINWEIS**
>
> Voreingestellt steht die serielle Schnittstelle dem Anwender nicht zur Verfügung, da sie für den Programm-Download und das Debugging genutzt wird.
>
> Setzt der Anwender das Systemmerkerbit SERIAL_MODE=TRUE, dann kann die Schnittstelle frei genutzt werden. Ein Debugging des Anwendungsprogramms ist dann nur noch über eine der 4 CAN-Schnittstellen oder über USB möglich.

Parameter der Eingänge

312

Parameter	Datentyp	Beschreibung
CLEAR	BOOL	TRUE: Empfangspuffer löschen FALSE: Funktion wird nicht ausgeführt

Parameter der Ausgänge

12931

Parameter	Datentyp	Beschreibung
RX	BYTE	empfangene Byte-Daten aus dem Empfangspuffer
AVAILABLE	WORD	Anzahl der empfangenen Bytes, die sich im Empfangspuffer befinden VOR dem Aufruf des FBs: 0 = keine Daten empfangen 1...1 000 = Anzahl von Bytes im Empfangspuffer
OVERFLOW	BOOL	TRUE: Überlauf des Datenpuffers ⇨ Datenverlust! FALSE: Datenpuffer ist ohne Datenverlust

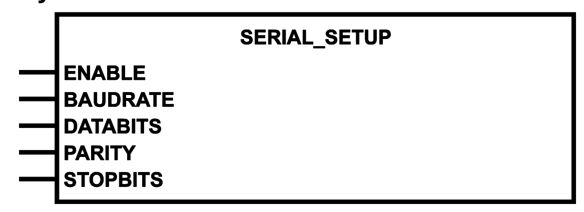
SERIAL_SETUP

302

Baustein-Typ = Funktionsbaustein (FB)

Baustein ist enthalten in Bibliothek `ifm_CR7132_Vxxyyzz.LIB`

Symbol in CODESYS:



Beschreibung

13000

SERIAL_SETUP initialisiert die serielle RS232-Schnittstelle.

Der FB muss nicht zwingend ausgeführt werden, um die serielle Schnittstelle verwenden zu können. Ohne FB-Aufruf gelten die folgend angegebenen Voreinstellungen.

Mit ENABLE=TRUE für einen Zyklus setzt der FB die serielle Schnittstelle auf die angegebenen Parameter. Die mit dem FB vorgenommenen Änderungen werden remanent gespeichert.

> **HINWEIS**
>
> Voreingestellt steht die serielle Schnittstelle dem Anwender nicht zur Verfügung, da sie für den Programm-Download und das Debugging genutzt wird.
>
> Setzt der Anwender das Systemmerkerbit SERIAL_MODE=TRUE, dann kann die Schnittstelle frei genutzt werden. Ein Debugging des Anwendungsprogramms ist dann nur noch über eine der 4 CAN-Schnittstellen oder über USB möglich.

Parameter der Eingänge

13002

Parameter	Datentyp	Beschreibung
ENABLE	BOOL	TRUE (nur 1 Zyklus lang): Schnittstelle initialisieren FALSE: im weiteren Programmablauf
BAUDRATE	DWORD	Baudrate zulässige Werte → Datenblatt Voreinstellwert → Datenblatt
DATABITS	BYTE := 8	Anzahl der Daten-Bits zulässig = 7 oder 8
PARITY	BYTE := 0	Parität zulässig: 0=keine, 1=gerade, 2=ungerade Falls DATABITS = 7 und PARITY = 0 parametriert: dann arbeitet der FB mit PARITY = 1
STOPBITS	BYTE := 1	Anzahl der Stopp-Bits zulässig = 1 oder 2

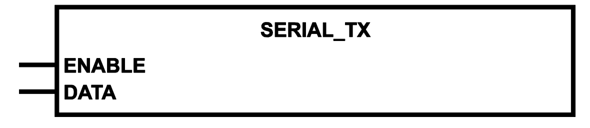
SERIAL_TX

296

Baustein-Typ = Funktionsbaustein (FB)

Baustein ist enthalten in Bibliothek `ifm_CR7132_Vxxyyzz.LIB`

Symbol in CODESYS:



Beschreibung

13003

SERIAL_TX überträgt ein Datenbyte über die serielle RS232-Schnittstelle.

Mit dem Eingang ENABLE kann die Übertragung freigegeben oder gesperrt werden.

Die SERIAL-Bausteine bilden die Grundlage für die Erstellung eines anwendungsspezifischen Protokolls für die serielle Schnittstelle.

> **HINWEIS**
>
> Voreingestellt steht die serielle Schnittstelle dem Anwender nicht zur Verfügung, da sie für den Programm-Download und das Debugging genutzt wird.
>
> Setzt der Anwender das Systemmerkerbit SERIAL_MODE=TRUE, dann kann die Schnittstelle frei genutzt werden. Ein Debugging des Anwendungsprogramms ist dann nur noch über eine der 4 CAN-Schnittstellen oder über USB möglich.

Parameter der Eingänge

300

Parameter	Datentyp	Beschreibung
ENABLE	BOOL	TRUE: Baustein ausführen FALSE: Baustein wird nicht ausgeführt > Baustein-Eingänge sind nicht aktiv > Baustein-Ausgänge sind nicht spezifiziert
DATA	BYTE	zu übertragender Wert

6.2.9 Bausteine: Eingangswerte verarbeiten

Inhalt

1602
1302

Hier zeigen wir Ihnen **ifm**-Funktionsbausteine zum Lesen und Verarbeiten der analogen oder binären Signale am Geräte-Eingang.

HINWEIS

Die in der Steuerungskonfiguration von CODESYS erscheinenden analogen Rohwerte kommen direkt aus dem ADC. Sie sind noch nicht korrigiert!

Deshalb können in der Steuerungskonfiguration bei gleichen Geräten unterschiedliche Rohwerte erscheinen.

Erst durch die **ifm**-FBs findet eine Fehlerkorrektur und Normierung statt. Die FBs liefern den korrigierten Wert.

INPUT_ANALOG

12929

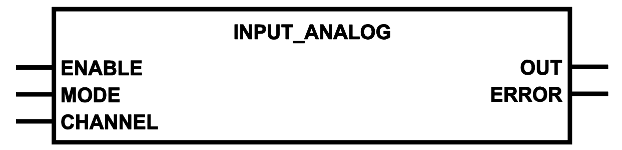
Baustein-Typ = Funktionsbaustein (FB)

Baustein ist enthalten in Bibliothek `ifm_CR7132_Vxxyyzz.LIB`

Für Sicherheitssignale zusätzlich ***SF_EQUIVALENT*** (→ Seite 283) oder ***SF_EQUIVALENT_WORD*** (→ Seite 287) einsetzen!

Für die Extended-Seite des ExtendedControllers endet der FB-Name mit "_E".

Symbol in CODESYS:



Beschreibung

12913
12916

INPUT_ANALOG ermöglicht Strom- und Spannungsmessung an den Analogkanälen (für alle Eingänge zugelassen).

Der FB liefert den aktuellen Analogwert am gewählten Analogkanal. Die Analogwerte werden normiert ausgegeben. Gleichzeitig werden die unkalibrierten Rohwerte über die Systemmerker ANALOGxx ausgegeben.

► Für Frequenz- und Periodenmessungen sowie Zählerfunktionen: MODE=1 (= IN_DIGITAL_H) einstellen!

Die Messung und der Ausgangswert resultieren aus der über MODE angegebenen Betriebsart:

12917

MODE dez	MODE hex	Eingang Betriebsart	FB-Ausgang OUT	Einheit
0	00	deaktiviert	---	---
1	01	Binäreingang, minus-schaltend (BH)	0 / 1	---
2	02	Binäreingang, plus-schaltend (BL)	0 / 1	---
4	04	Stromeingang	0…20 000	µA
8	08	Spannungseingang	0…10 000	mV
16	10	Spannungseingang	0…32 000	mV
32	20	Spannungseingang, ratiometrisch	0…1 000	‰

Diese und weitere Betriebsarten der Eingänge auch einstellbar mit...

SET_INPUT_MODE (→ Seite 264)	weist einem Eingangskanal eine Betriebsart zu

13020

HINWEIS

Nach dem Umschalten in einen anderen Modus während der Laufzeit dauert es wenige Zyklen, bis der Ausgangswert wieder korrekt ist.

Wenn derselbe Eingangskanal während der Laufzeit unterschiedlich konfiguriert wurde, dann gilt die zuletzt vorgenommene Konfiguration.

ABER:

Falls der Eingangskanal als Sicherheitseingang konfiguriert ist (SAFETY=TRUE):

Wenn derselbe Eingangskanal während der Laufzeit unterschiedlich konfiguriert wurde, dann wertet die Steuerung dies als schweren Fehler!

Beispiele:
- zuerst 'safety', dann 'non-safety'
- zuerst 'IN_DIGITAL_H', dann 'IN_VOLTAGE_10'

Bei den folgenden Betriebsarten werden auch Werte über dem nominalen Wertebereich ausgegeben, soweit es die Hardware ermöglicht:
- IN_CURRENT
- IN_VOLTAGE
- IN_RATIO

Bei der Betriebsart IN_CURRENT ist eine Überstromüberwachung immer aktiv.
Bei einem Eingangsstrom > 21,7 mA für > 66 ms schaltet die Überwachung die Messbürde für eine Sekunde weg und signalisiert über den entsprechenden Merker einen Fehler.

Parameter der Eingänge

12924

Parameter	Datentyp	Beschreibung
ENABLE	BOOL	TRUE: Baustein ausführen FALSE: Baustein wird nicht ausgeführt > Baustein-Eingänge sind nicht aktiv > Baustein-Ausgänge sind nicht spezifiziert
MODE	WORD	Betriebsart des Eingangskanals: 0 = 0x0000 IN_NOMODE (Aus; Voreinstellung aktiv) 1 = 0x0001 IN_DIGITAL_H voreingestellt 2 = 0x0002 IN_DIGITAL_L 4 = 0x0004 IN_CURRENT 0...20 000 µA 8 = 0x0008 IN_VOLTAGE10 0...10 000 mV 16 = 0x0010 IN_VOLTAGE30 0...30 000 mV 32 = 0x0020 IN_RATIO 0...1 000 ‰
CHANNEL	BYTE	Nummer des Eingangskanals (0...15) 0...15 für die Eingänge I00...I15 Für den FB xxx_**E** (falls vorhanden) gilt: 0...15 für die Eingänge I00_E...I15_E

Parameter der Ausgänge

12925

Parameter	Datentyp	Beschreibung
OUT	WORD	Ausgangswert entsprechend MODE bei ungültiger Einstellung: OUT = "0"
ERROR	DWORD	Fehler-Code aus diesem FB-Aufruf → ***Fehler-Codes*** (→ Seite 392) (mögliche Meldungen → folgende Tabelle)

Mögliche Ergebnisse für ERROR (n=beliebiger Wert):

Der 32-Bit-Fehler-Code besteht aus vier 8-Bit-Werten (DWORD).

4. Byte	3. Byte	2. Byte	1. Byte
Fehlerklasse	anwendungsspezifischer Fehler-Code	Fehlerquelle	Fehlerursache

Wert [hex]	Beschreibung
00 00 00 00	kein Fehler
02 00 00 F8	falscher Parameter ⇨ schwerer Fehler

weitere mögliche Fehler-Codes → Tabellen im Kapitel ***Fehlerklasse (4. Byte)*** (→ Seite 397)
- Eintrag in Spalte [Funktionsbaustein] = INPUT_ANALOG
- Eintrag in Spalte [Funktionsbaustein] = INPUT_ANALOG_E

SET_INPUT_MODE

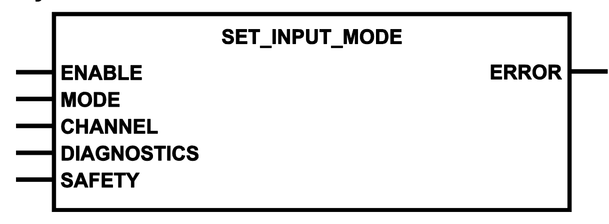
13015

Baustein-Typ = Funktionsbaustein (FB)

Baustein ist enthalten in Bibliothek `ifm_CR7132_Vxxyyzz.LIB`

Für die Extended-Seite des ExtendedControllers endet der FB-Name mit "_E".

Symbol in CODESYS:



Beschreibung

13017
11944

Mit SET_INPUT_MODE können Sie den Eingangskanälen Betriebsarten zuweisen.
→ Kapitel ***Mögliche Betriebsarten Ein-/Ausgänge*** (→ Seite 436)

Im laufenden Betrieb sollte die Betriebsart nicht geändert werden.

Auch mit dem FB ***INPUT_ANALOG*** (→ Seite 261) kann die Betriebsart an einem Eingang konfiguriert werden.

18414

Falls Eingang I15 nicht verwendet:
- ▶ Eingang I15 als Binäreingang konfigurieren!

13020

HINWEIS

Nach dem Umschalten in einen anderen Modus während der Laufzeit dauert es wenige Zyklen, bis der Ausgangswert wieder korrekt ist.

Wenn derselbe Eingangskanal während der Laufzeit unterschiedlich konfiguriert wurde, dann gilt die zuletzt vorgenommene Konfiguration.

ABER:

Falls der Eingangskanal als Sicherheitseingang konfiguriert ist (SAFETY=TRUE):

Wenn derselbe Eingangskanal während der Laufzeit unterschiedlich konfiguriert wurde, dann wertet die Steuerung dies als schweren Fehler!

Beispiele:
- zuerst 'safety', dann 'non-safety'
- zuerst 'IN_DIGITAL_H', dann 'IN_VOLTAGE_10'

Falls der Eingangskanal als Sicherheitseingang konfiguriert ist (SAFETY=TRUE):
- ▶ DIAGNOSTICS = FALSE einstellen!
  TRUE gilt nur für Namur-Sensoren.
- ▶ MODE = 1 einstellen (IN_DIGITAL_H).

- Bei den folgenden Betriebsarten werden auch Werte über dem nominalen Wertebereich ausgegeben, soweit es die Hardware ermöglicht:
  • IN_CURRENT
  • IN_VOLTAGE
  • IN_RATIO
- Bei der Betriebsart IN_CURRENT ist eine Überstromüberwachung immer aktiv.
  Bei einem Eingangsstrom > 21,7 mA für > 66 ms schaltet die Überwachung die Messbürde für eine Sekunde weg und signalisiert über den entsprechenden Merker einen Fehler.

Parameter der Eingänge

13022

Parameter	Datentyp	Beschreibung
ENABLE	BOOL	FALSE ⇨ TRUE (Flanke): Baustein initialisieren (nur 1 Zyklus) > Baustein-Eingänge lesen TRUE: Baustein ausführen FALSE: Baustein wird nicht ausgeführt > Baustein-Eingänge sind nicht aktiv > Baustein-Ausgänge sind nicht spezifiziert
MODE	WORD	Betriebsart des Eingangskanals CHANNEL: 0 = 0x0000 IN_NOMODE (Aus; Voreinstellung aktiv) 1 = 0x0001 IN_DIGITAL_H (voreingestellt) 2 = 0x0002 IN_DIGITAL_L 4 = 0x0004 IN_CURRENT 0...20 000 µA 8 = 0x0008 IN_VOLTAGE10 0...10 000 mV 16 = 0x0010 IN_VOLTAGE30 0...30 000 mV 32 = 0x0020 IN_RATIO 0...1 000 ‰
CHANNEL	BYTE	Nummer des Eingangskanals (0...15) 0...15 für die Eingänge I00...I15 Für den FB xxx_**E** (falls vorhanden) gilt: 0...15 für die Eingänge I00_E...I15_E
DIAGNOSTICS	BOOL	TRUE: Kanal mit Diagnosefunktion nur wirksam für IN_DIGITAL_H Fehlermeldungen: • Leiterbruch oder Schluss gegen Masse bei Eingangsspannung < 1V für > 66 ms • Schluss gegen Versorgung bei Eingangsspannung > 95 % VBBS für > 66 ms FALSE: Kanal ohne Diagnosefunktion
SAFETY	BOOL	CHANNEL soll als Sicherheitskanal betrieben werden Nicht zulässig für folgende Betriebsarten: • IN_DIGITAL_L • Eingang erfasst Impulssignale via FB • Eingang erfasst Signale eines SafetySwitch Sobald SAFETY=TRUE, ist Folgendes für diesen Kanal nicht mehr zulässig: • SAFETY=FALSE • ändern von MODE Andernfalls ⇨ schwerer Fehler! TRUE: CHANNEL ist Sicherheitskanal FALSE: CHANNEL ist Standardkanal

Parameter der Ausgänge

13025

Parameter	Datentyp	Beschreibung
ERROR	DWORD	Fehler-Code aus diesem FB-Aufruf → ***Fehler-Codes*** (→ Seite 392) (mögliche Meldungen → folgende Tabelle)

Mögliche Ergebnisse für ERROR (n=beliebiger Wert):

Der 32-Bit-Fehler-Code besteht aus vier 8-Bit-Werten (DWORD).

4. Byte	3. Byte	2. Byte	1. Byte
Fehlerklasse	anwendungsspezifischer Fehler-Code	Fehlerquelle	Fehlerursache

Wert [hex]	Beschreibung
00 00 00 00	kein Fehler
02 00 00 F8	falscher Parameter ⇨ schwerer Fehler

6.2.10 Bausteine: Eingangswerte sicher verarbeiten

Inhalt

12699

Hier finden Sie Funktionsbausteine zum sichern Verarbeiten von sicherheitsrelevanten Eingangswerten.

8340

⚠ WARNUNG

Für die sichere Funktion der Anwendungsprogramme, die vom Anwender erstellt werden, ist dieser selbst verantwortlich. Bei Bedarf muss er zusätzlich entsprechend der nationalen Vorschriften eine Abnahme durch entsprechende Prüf- und Überwachungsorganisationen durchführen lassen.

Wo Sicherheitsgeräte 2-kanalig verarbeitet werden müssen, gelten folgende Regeln:

- Die Eingangspaare der Sicherheitsgeräte müssen in verschiedenen Eingangsblöcken liegen. Beispiel: Eingang 1 im Bereich I00...I07, Eingang 2 im Bereich I08...I15
- Die Werte dieser Eingangspaare müssen mit 2-kanaligen Sicherheits-FBs validiert werden (z.B. SF_EQUIVALENT). Der Sicherheitsausgang dieser FBs bildet den Sicherheitseingang des folgenden 1-kanaligen Sicherheits-FBs (z.B. SF_EMERGENCYSTOP).

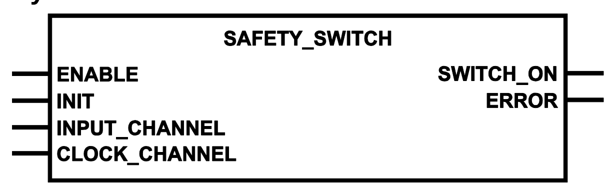
SAFETY_SWITCH

12955

Baustein-Typ = Funktionsbaustein (FB)

Baustein ist enthalten in Bibliothek `ifm_CR7132_Vxxyyzz.LIB`

Symbol in CODESYS:



Beschreibung

11783

SAFETY_SWITCH dient zum Betrieb der 1-kanaligen SafetySwitch der **ifm electronic gmbh**.

Die Konfiguration des FBs (gewählter Eingangskanal und Taktausgang) wird bei der Initialisierung übernommen und somit fest eingestellt. Während des Programmablaufs kann die Konfiguration des Funktionsbausteins nicht geändert werden.

ACHTUNG

Gefahr, dass nicht alle Fehler erkannt werden oder dass ständig Fehler gemeldet werden.

Bei einem Querschluss zu einem anderen Signal kann es vorkommen, dass zeitweise kein Fehler gemeldet wird, wenn der erfasste Signalpegel dem erwarteten Signalpegel entspricht.

► Für in INPUT_CHANNEL definierten Eingang **keinen** Glitch-Filter (Ixx_DFILTER) aktivieren!

► Mit ENABLE=TRUE den FB aktivieren.

> Der SafetyController sendet am mit CLOCK_CHANNEL definierten Ausgangskanal ein Taktsignal an den angeschlossenen Sensor.

> Der Sensor gibt das Taktsignal leicht zeitverzögert wieder an den Controller zurück, solange sich ein Bedämpfungselement in seiner Freigabezone befindet.

> Der SafetyController empfängt das vom Sensor ausgegebene Signal am mit INPUT_CHANNEL definierten Eingang und prüft das Signal auf Korrektheit.

> Verwenden mehrere INPUT_CHANNEL denselben CLOCK_CHANNEL als Signalquelle, dann gilt für die Berechnung der High-Phase des Ausgangssignals der niedrigste verwendete INPUT_CHANNEL.

Max. 9 SafetySwitch dürfen in Reihe kaskadiert werden.

Um einen Querschluss zwischen mehreren SafetySwitch-Ketten zu erkennen, werden die Signale an den unterschiedlichen Ausgängen mit unterschiedlicher Periodendauer ausgegeben:

Ausgangssignal	Dauer	Toleranz
FALSE	52 ms	+ 4 ms
TRUE	55 ms + (5 ms • INPUT_CHANNEL)	+ 4 ms

> Wenn der SafetyController das erzeugte Taktsignal mit dem richtigen zeitlichen Ablauf am konfigurierten Eingang liest, wird bei Bedämpfung des Sensors der Ausgang SWITCH_ON auf TRUE gesetzt.

Das sichere Ausgangssignal SWITCH_ON kann im Anwendungsprogramm ausgewertet werden.

ACHTUNG

> Das Ergebnis kann in jedem SPS-Zyklus wechseln, wenn die Signal-Konstellation dies ergibt.
▶ Der Programmierer muss einen gemeldeten Fehler im selben Zyklus auswerten.
▶ Im Fehlerfall muss der Programmierer die Maschine / Anlage in den sicheren Zustand bringen.

Als Fehler erkannt werden folgende Vorgänge:

- beim Deaktivieren des Taktsignals eine Verzögerung vom Ausgang zum Eingang von > 14,85 ms + 4 ms Toleranz
- beim Aktivieren des Taktsignals eine Verzögerung vom Ausgang zum Eingang von < 0,7 ms
- wenn Eingang=TRUE, obwohl der Taktausgang=FALSE (unter Berücksichtigung der Verzögerungszeit im ersten Punkt)

Vor dem Zurücksetzen des Fehlers:
▶ Eine gewisse Zeit lang (z. B. 300 ms) prüfen, ob der Fehler nicht mehr anliegt.

Parameter der Eingänge

12950

Parameter	Datentyp	Beschreibung
ENABLE	BOOL	TRUE: Baustein ausführen FALSE: Baustein wird nicht ausgeführt > Baustein-Eingänge sind nicht aktiv > Baustein-Ausgänge sind nicht spezifiziert
INIT	BOOL	TRUE (nur 1 Zyklus lang): Baustein und Schnittstelle werden initialisiert FALSE: im weiteren Programmablauf
INPUT_CHANNEL	BYTE	Nummer des Eingangskanals zulässig = 0...7
CLOCK_CHANNEL	BYTE	Nummer des Ausgangskanals für das Taktsignal zulässig = 0...15 Der definierte Taktausgang darf NICHT als Sicherheitsausgang und NICHT als PWM-Ausgang konfiguriert werden! Derselbe Ausgang darf in mehreren Instanzen dieses FBs als Taktausgang verwendet werden.

Parameter der Ausgänge

12957

Parameter	Datentyp	Beschreibung
SWITCH_ON	BOOL	Zustand des sicheren Sensors: TRUE: Sensorausgang geschaltet (Bedämpfungselement innerhalb der Freigabezone) FALSE (= Initialwert): Sensorausgang ausgeschaltet (sicherer Zustand)
ERROR	DWORD	Fehler-Code aus diesem FB-Aufruf → ***Fehler-Codes*** (→ Seite 392) (mögliche Meldungen → folgende Tabelle)

Mögliche Ergebnisse für ERROR (n=beliebiger Wert):

Der 32-Bit-Fehler-Code besteht aus vier 8-Bit-Werten (DWORD).

4. Byte	3. Byte	2. Byte	1. Byte
Fehlerklasse	anwendungsspezifischer Fehler-Code	Fehlerquelle	Fehlerursache

Wert [hex]	Beschreibung
00 00 00 00	kein Fehler
00 00 00 2A	Verzögerung zu klein (erkannt bei CLOCK=HIGH)
00 00 00 33	Input=TRUE, aber keine Flanke erkannt eventuell "stuck at 1" (erkannt bei CLOCK=HIGH)
00 00 00 34	zu lange oder erneut Input=TRUE (erkannt bei CLOCK=LOW)
00 00 00 4C	falscher Parameter am FB

weitere mögliche Fehler-Codes → Tabellen im Kapitel ***Fehlerklasse (4. Byte)*** (→ Seite 397)
• Eintrag in Spalte [Funktionsbaustein] = SAFETY_SWITCH

Im Fehlerfall meldet ***SHOW_ERROR_LIST*** (→ Seite 389) den Fehler-Code 0x0200nn0B:
0x02 = schwerer Fehler
0x0B = Safety-Diagnose SAFETY_SWITCH
nn = Nr des Eingangs, z.B. 0x17 für Eingang I07

Legende zu den Ein- und Ausgängen der Sicherheits-FBs 'SF_...'

12700

- Alle Ein- und Ausgänge der Sicherheits-FBs 'SF_...' werden beim Hochlaufen (Booten) der Steuerung mit Initialwerten belegt. So werden undefinierte Zustände der FBs verhindert. Die Initialwerte führen immer zum sicheren Zustand der Steuerung.
- Alle Sicherheits-FBs arbeiten nur, wenn der Eingang ENABLE=TRUE ist.
- Der Ausgang READY=TRUE gibt an, dass der FB aktiv ist.
- Der Ausgang SAFETYDEMAND=TRUE signalisiert, dass Handlungsbedarf durch den Bediener erforderlich ist.
- Der Eingang S_SAFETYACTIVE=TRUE signalisiert dem FB, dass der relevante Prozess im Sicherheitsmodus ist.
- Der Ausgang RESETREQUEST=TRUE fordert vom Bediener:
  • gegebenenfalls die falschen Eingangssignale beheben
  • mit RESET-Signal bestätigen, dass der FB den sicheren Prozess fortsetzen darf
  Andernfalls verbleibt der FB im Warten- oder Fehler-Zustand.
  ℹ FBs ohne Eingang RESET: der Ausgang RESETREQUEST ist ohne Funktion.
- Der Eingang RESET=TRUE (nur 1 Zyklus) signalisiert dem FB:
  • den sicheren Prozess fortsetzen, sofern die Fehler beseitigt sind.
  Dauerhafter RESET führt zum Fehler.
- Der Ausgang ERROR=TRUE signalisiert einen Fehler.
  Der Fehler-Code erscheint im Ausgang DIAGCODE mit Werten ab 0xC000.
- Der Ausgang DIAGCODE liefert permanent Informationen über den Status des FBs.

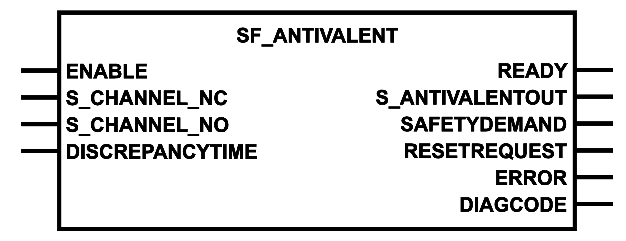
SF_ANTIVALENT

12484

Baustein-Typ = Funktionsbaustein (FB)

Baustein ist enthalten in Bibliothek `ifm_SafetyPLCopen_Vxxyyzz.lib`

Symbol in CODESYS:



Beschreibung

12486

SF_ANTIVALENT vergleicht zwei binäre sichere Eingänge (NC und NO) miteinander.
Der FB überwacht den Zeitabstand zwischen den Flankenwechseln auf beiden Eingängen.
• Einer der Eingänge (NC) überwacht den Öffnerkontakt (normally closed).
• Der andere Eingang (NO) überwacht den Schließerkontakt (normally open).

15749

> ⓘ Sind die Eingänge des FB mit einem OSSD-Sicherheitsschalter belegt, dann den FB-Eingang DISCREPANCYTIME mit einem Wert ≥ 10 ms einstellen (je nach Applikation).
> Andernfalls kann der FB fälschlich eine Fehlstellung der Eingänge melden.

Parameter der Eingänge

12487

Parameter	Datentyp	Beschreibung
ENABLE	BOOL	TRUE: Baustein ausführen FALSE: Baustein wird nicht ausgeführt > Baustein-Eingänge sind nicht aktiv > Baustein-Ausgänge sind nicht spezifiziert
S_CHANNEL_NC	BOOL	binärer sicherheitsrelevanter Öffner-Eingang TRUE: Kontakt geschlossen FALSE (= Initialwert): Kontakt offen
S_CHANNEL_NO	BOOL	binärer sicherheitsrelevanter Schließer-Eingang TRUE (= Initialwert): Kontakt geschlossen FALSE: Kontakt offen
DISCREPANCYTIME	TIME	Maximal zulässige Zeitdifferenz zwischen den Flankenwechseln an den Eingängen NC und NO von FALSE nach TRUE oder von TRUE nach FALSE. Initialwert = T#0ms

Parameter der Ausgänge

12488

Parameter	Datentyp	Beschreibung
READY	BOOL	TRUE: Funktionsbaustein ist aktiv Die Werte an den FB-Ausgängen sind gültig FALSE (= Initialwert): Funktionsbaustein ist nicht aktiv
S_ANTIVALENTOUT	BOOL	Sicherheitsausgang TRUE: S_CHANNEL_NC=TRUE **und** S_CHANNEL_NO=FALSE **und** die letzten Flankenwechsel auf den Eingängen waren innerhalb der Überwachungszeit FALSE (= Initialwert): S_CHANNEL_NC=FALSE **oder** S_CHANNEL_NO=TRUE **oder** ein Flankenwechsel war außerhalb der Überwachungszeit
SAFETYDEMAND	BOOL	TRUE: Sicherheitsfunktion ist angefordert Aktion des Maschinenführers erforderlich FALSE (= Initialwert): Sicherheitsfunktion nicht angefordert
RESETREQUEST	BOOL	(FB-Ausgang hier ohne Funktion)
ERROR	BOOL	TRUE: ein Fehler ist aufgetreten FB ist im Fehlerzustand FALSE (= Initialwert): kein Fehler aufgetreten
DIAGCODE	WORD	Informationen zum aktuellen Zustand des FBs (mögliche Meldungen → folgende Tabelle)

Mögliche Ergebnisse für DIAGCODE:

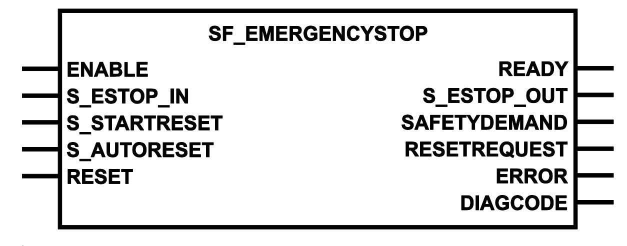
Wert [hex]	Beschreibung
0000	(Initialwert) Funktionsbaustein ist nicht aktiv
8000	Sicherheitsausgang ist freigegeben
8001	Funktionsbaustein ist bereit: READY=TRUE kein Sicherheitseingang ist aktiv
8004	Sicherheitseingang 1 ist aktiv Sicherheitseingang 2 ist noch nicht aktiv Überwachungszeit läuft
8005	einer der beiden Sicherheitseingänge ist noch aktiv der andere Sicherheitseingang ist nicht mehr aktiv Überwachungszeit läuft
8014	Sicherheitseingang 1 ist noch nicht aktiv Sicherheitseingang 2 ist aktiv Überwachungszeit läuft
C001	Überwachungszeit abgelaufen (FB wartet auf Kanal 2)
C002	Überwachungszeit abgelaufen (FB wartet auf Kanal 1)
C003	Überwachungszeit abgelaufen (FB wartet auf beide Kanäle=inaktiv)

SF_EMERGENCYSTOP 12514

Baustein-Typ = Funktionsbaustein (FB)

Baustein ist enthalten in Bibliothek `ifm_SafetyPLCopen_Vxxyyzz.lib`

Symbol in CODESYS:



Beschreibung 12517

Der FB SF_EMERGENCYSTOP überwacht einen Not-Halt-Schalter.

HINWEIS

Falls die geforderte Sicherheitsstufe anders nicht erreicht werden kann:

► Zur Überwachung von zweikanaligen Sicherheitseinrichtungen (z.B. NOT-HALT-Schalter) ***SF_EQUIVALENT*** (→ Seite 283) oder ***SF_ANTIVALENT*** (→ Seite 272) einsetzen!

Falls die damit aufgebaute Sicherheitseinrichtung nicht regelmäßig genutzt wird:

► Sicherheitseinrichtung in festgelegten Zeitabständen manuell testen!

> So kann ein Fehler (z.B. in der Verkabelung oder im Schalter) rechtzeitig erkannt werden.

Ablauf, wenn S_AUTORESET=FALSE und S_STARTRESET=FALSE:

► Nach dem Start des FBs den Not-Halt-Schalter entriegeln ⇨ S_ESTOP_IN=TRUE.

► Zur Prüfung der Schaltung einmalig kurz RESET setzen (1 Zyklus).

> Der Sicherheitsausgang wird aktiv: S_ESTOP_OUT=TRUE.

► Nach jedem Auslösen von NOT-HALT:
- NOT-HALT-Schalter entriegeln
- einmalig kurz RESET setzen
- Sicherheitsausgang wird aktiv

Ablauf, wenn S_AUTORESET=FALSE und S_STARTRESET=TRUE:

► Den NOT-HALT-Schalter entriegeln ⇨ S_ESTOP_IN=TRUE.

► Mit dem Start des FBs (ENABLE=TRUE) wird sofort S_ESTOP_OUT=TRUE.

► Nach jedem Auslösen von NOT-HALT:
- NOT-HALT-Schalter entriegeln
- einmalig kurz RESET setzen
- Sicherheitsausgang wird aktiv

Ablauf, wenn S_AUTORESET=TRUE und S_STARTRESET=FALSE:

► Nach dem Start des FBs den NOT-HALT-Schalter entriegeln ⇨ S_ESTOP_IN=TRUE.
► Zur Prüfung der Schaltung einmalig kurz RESET setzen (1 Zyklus).
> Der Sicherheitsausgang wird aktiv: S_ESTOP_OUT=TRUE.
> Ab jetzt arbeitet der Sicherheitsausgang in direkter Abhängigkeit des NOT-HALT-Schalters:
S_ESTOP_IN=TRUE ⇨ S_ESTOP_OUT=TRUE
S_ESTOP_IN=FALSE ⇨ S_ESTOP_OUT=FALSE

Parameter der Eingänge

12518

Parameter	Datentyp	Beschreibung
ENABLE	BOOL	TRUE: Baustein ausführen FALSE: Baustein wird nicht ausgeführt > Baustein-Eingänge sind nicht aktiv > Baustein-Ausgänge sind nicht spezifiziert
S_ESTOP_IN	BOOL	NOT-HALT-Eingangssignal TRUE: NOT-HALT-Schalter wurde nicht betätigt FALSE (= Initialwert): NOT-HALT-Schalter wurde betätigt
S_STARTRESET	BOOL	Nach dem Aktivieren des FB ... TRUE: ... erfolgt automatisch ein Reset. FALSE (= Initialwert): ... ist ein manueller Reset erforderlich.
S_AUTORESET	BOOL	Nach dem Zurücksetzen des Sicherheitsschalters ... TRUE: ... erfolgt automatisch ein Reset. FALSE (= Initialwert): ... ist ein manueller Reset erforderlich.
RESET	BOOL	TRUE (nur 1 Zyklus lang): • Bestätigung: sicherer Zustand ist erfüllt • Bestätigung: Fehler ist behoben sonst: diese Funktion wird nicht ausgeführt (=Initialwert)

Parameter der Ausgänge

12519

Parameter	Datentyp	Beschreibung
READY	BOOL	TRUE: Funktionsbaustein ist aktiv Die Werte an den FB-Ausgängen sind gültig FALSE (= Initialwert): Funktionsbaustein ist nicht aktiv
S_ESTOP_OUT	BOOL	TRUE: Sicherheitsausgang ist aktiv FALSE (= Initialwert): Sicherheitsausgang ist deaktiviert: • NOT-HALT-Schalter ist betätigt, • ein Reset ist erforderlich • oder ein interner Fehler liegt an
SAFETYDEMAND	BOOL	TRUE: Sicherheitsfunktion ist angefordert Aktion des Maschinenführers erforderlich FALSE (= Initialwert): Sicherheitsfunktion nicht angefordert
RESETREQUEST	BOOL	Ein Reset ist erforderlich, damit der FB weiterarbeiten kann TRUE: Reset ist erforderlich FALSE (= Initialwert): kein Reset erforderlich
ERROR	BOOL	TRUE: ein Fehler ist aufgetreten FB ist im Fehlerzustand FALSE (= Initialwert): kein Fehler aufgetreten
DIAGCODE	WORD	Informationen zum aktuellen Zustand des FBs (mögliche Meldungen → folgende Tabelle)

Mögliche Ergebnisse für DIAGCODE:

Wert [hex]	Beschreibung
0000	(Initialwert) Funktionsbaustein ist nicht aktiv
8000	Sicherheitsausgang ist freigegeben
8001	Funktionsbaustein ist bereit: READY=TRUE prüfen, ob S_STARTRESET benötigt wird
8002	Funktionsbaustein ist bereit: READY=TRUE prüfen, ob RESET=FALSE FB wartet auf erstmalig Sicherheitseingang=TRUE
8003	erstmalig Sicherheitseingang=TRUE FB wartet auf RESET=TRUE
8004	Funktionsbaustein ist bereit: READY=TRUE prüfen, ob RESET=FALSE FB wartet auf Sicherheitseingang=TRUE
8005	Sicherheitseingang=TRUE FB wartet auf RESET=TRUE oder auf S_AUTORESET=TRUE
C001	Funktionsbaustein ist bereit: READY=TRUE RESET=TRUE FB wartet auf erstmalig Sicherheitseingang=TRUE
C002	Funktionsbaustein ist bereit: READY=TRUE RESET=TRUE FB wartet auf Sicherheitseingang=TRUE

SF_ENABLESWITCH

12542

Baustein-Typ = Funktionsbaustein (FB)

Baustein ist enthalten in Bibliothek `ifm_SafetyPLCopen_Vxxyyzz.lib`

Symbol in CODESYS:

SF_ENABLESWITCH	
ENABLE	READY
S_SAFETYACTIVE	S_ENABLESWITCH_OUT
S_ENABLESWITCH_CH1	SAFETYDEMAND
S_ENABLESWITCH_CH2	RESETREQUEST
S_AUTORESET	ERROR
RESET	DIAGCODE

Beschreibung

12544

SF_ENABLESWITCH dient zum Auswerten der Signale einer Freigabetaste mit drei Schaltstufen.

In kritischen Anwendungen dürfen bei geöffneter Schutztür manuell ausgelöste Bewegungen der Maschine nur starten, wenn zusätzlich zum Bewegungskommando eine Freigabetaste (= Zustimmtaste) betätigt wird. Funktionsweise:

Freigabetaste	Funktion	E1, E2 CH1	E3, E4 CH2	Auswirkung an der Maschine
losgelassen (= erste Stufe)	keine Freigabe	FALSE	TRUE	• Bewegungen werden gestoppt
zweite Stufe gedrückt	Freigabe	TRUE	TRUE	• manuell ausgelöste Bewegungen sind möglich, solange Freigabetaste gedrückt bleibt
dritte Stufe gedrückt	Panikschaltung, NOT-HALT	FALSE	FALSE	• Bewegungen werden gestoppt • NOT-HALT-Funktion
dritte Stufe losgelassen, zweite Stufe gedrückt	keine Freigabe	FALSE	TRUE	• Bewegungen werden gestoppt • Die Freigabetaste ganz loslassen vor erneutem Betätigen!

Parameter der Eingänge

12546

Parameter	Datentyp	Beschreibung
ENABLE	BOOL	TRUE: Baustein ausführen FALSE: Baustein wird nicht ausgeführt > Baustein-Eingänge sind nicht aktiv > Baustein-Ausgänge sind nicht spezifiziert
S_SAFETYACTIVE	BOOL	Die Maschine ist im Sicherheitsmodus: z.B. Bewegungsgeschwindigkeit ist sicher reduziert. Nur bei aktivem Sicherheitsmodus darf die Freigabetaste den Sicherheitsausgang aktivieren. TRUE: der Sicherheitsmodus ist aktiv Die Freigabetaste kann den Sicherheitsausgang aktivieren. FALSE (= Initialwert): der Sicherheitsmodus ist nicht aktiv
S_ENABLESWITCH_CH1	BOOL	Zustand der Kontakte E1 und E2 der angeschlossenen Freigabetaste TRUE: Kontakte sind geschlossen FALSE (= Initialwert): Kontakte sind offen
S_ENABLESWITCH_CH2	BOOL	Zustand der Kontakte E3 und E4 der angeschlossenen Freigabetaste TRUE: Kontakte sind geschlossen FALSE (= Initialwert): Kontakte sind offen
S_AUTORESET	BOOL	Nach dem Zurücksetzen des Sicherheitsschalters ... TRUE: ... erfolgt automatisch ein Reset. FALSE (= Initialwert): ... ist ein manueller Reset erforderlich.
RESET	BOOL	TRUE (nur 1 Zyklus lang): • Bestätigung: sicherer Zustand ist erfüllt • Bestätigung: Fehler ist behoben sonst: diese Funktion wird nicht ausgeführt (=Initialwert)

Parameter der Ausgänge

12547

Parameter	Datentyp	Beschreibung
READY	BOOL	TRUE: Funktionsbaustein ist aktiv Die Werte an den FB-Ausgängen sind gültig FALSE (= Initialwert): Funktionsbaustein ist nicht aktiv
S_ENABLESWITCH_OUT	BOOL	Sicherheitsausgang TRUE: Sicherheitsfunktion freigegeben FALSE (= Initialwert): Sicherheitsfunktion gesperrt
SAFETYDEMAND	BOOL	TRUE: Sicherheitsfunktion ist angefordert Aktion des Maschinenführers erforderlich FALSE (= Initialwert): Sicherheitsfunktion nicht angefordert
RESETREQUEST	BOOL	Ein Reset ist erforderlich, damit der FB weiterarbeiten kann TRUE: Reset ist erforderlich FALSE (= Initialwert): kein Reset erforderlich
ERROR	BOOL	TRUE: ein Fehler ist aufgetreten FB ist im Fehlerzustand FALSE (= Initialwert): kein Fehler aufgetreten
DIAGCODE	WORD	Informationen zum aktuellen Zustand des FBs (mögliche Meldungen → folgende Tabelle)

Mögliche Ergebnisse für DIAGCODE:

Wert [hex]	Beschreibung
0000	(Initialwert) Funktionsbaustein ist nicht aktiv
8000	Sicherheitsausgang ist freigegeben
8004	Funktionsbaustein ist bereit + Sicherheitsmodus ist nicht aktiv
8006	Funktionsbaustein ist bereit + Sicherheitsmodus ist aktiv
8007	Funktionsbaustein ist bereit + Sicherheitsmodus ist aktiv + NOT-HALT
C001	Überwachungszeit abgelaufen (FB wartet auf Kanal 2)
C002	Fehler im Ablauf, z.B. Reihenfolge FB wartet auf RESET
C010	Startbedingung für S_SAFETYACTIVE ist nicht erfüllt
C020	Funktionsbaustein ist bereit + Sicherheitsmodus ist aktiv FB wartet auf RESET
C030	Fehler im Ablauf Safety-Startbedingung erreichen! kein RESET ist erforderlich
C040	Fehler im Ablauf kein Safety-Betrieb FB wartet auf RESET

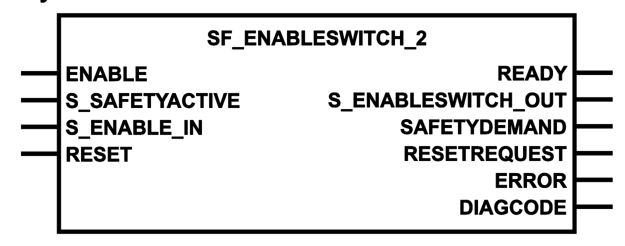
SF_ENABLESWITCH_2

12667

Baustein-Typ = Funktionsbaustein (FB)

Baustein ist enthalten in Bibliothek `ifm_SafetyPLCopen_Vxxyyzz.lib`

Symbol in CODESYS:



Beschreibung

12671

SF_ENABLESWITCH_2 dient zum Auswerten der Signale einer Freigabetaste mit zwei oder drei Schaltstufen.

In kritischen Anwendungen dürfen bei geöffneter Schutztür manuell ausgelöste Bewegungen der Maschine nur starten, wenn zusätzlich zum Bewegungskommando eine Freigabetaste (= Zustimmtaste) betätigt wird. Funktionsweise:

Freigabetaste	Funktion	S_ENABLE_IN	Auswirkung an der Maschine
losgelassen (= erste Stufe)	keine Freigabe	FALSE	• Bewegungen werden gestoppt
zweite Stufe gedrückt	Freigabe	TRUE	• manuell ausgelöste Bewegungen sind möglich, solange Freigabetaste gedrückt bleibt
dritte Stufe*) gedrückt	Panikschaltung, NOT-HALT	FALSE	• Bewegungen werden gestoppt • NOT-HALT-Funktion
dritte Stufe*) losgelassen, zweite Stufe gedrückt	keine Freigabe	FALSE	• Bewegungen werden gestoppt • Die Freigabetaste ganz loslassen vor erneutem Betätigen!

*) sofern dritte Schaltstufe vorhanden

Parameter der Eingänge

12627

Parameter	Datentyp	Beschreibung
ENABLE	BOOL	TRUE: Baustein ausführen FALSE: Baustein wird nicht ausgeführt > Baustein-Eingänge sind nicht aktiv > Baustein-Ausgänge sind nicht spezifiziert
S_SAFETYACTIVE	BOOL	Die Maschine ist im Sicherheitsmodus: z.B. Bewegungsgeschwindigkeit ist sicher reduziert. Nur bei aktivem Sicherheitsmodus darf die Freigabetaste den Sicherheitsausgang aktivieren. TRUE: der Sicherheitsmodus ist aktiv Die Freigabetaste kann den Sicherheitsausgang aktivieren. FALSE (= Initialwert): der Sicherheitsmodus ist nicht aktiv
S_ENABLE_IN	BOOL	Zustand der Kontakte der angeschlossenen Freigabetaste TRUE: Kontakte sind geschlossen FALSE (= Initialwert): Kontakte sind offen
RESET	BOOL	TRUE (nur 1 Zyklus lang): • Bestätigung: sicherer Zustand ist erfüllt • Bestätigung: Fehler ist behoben sonst: diese Funktion wird nicht ausgeführt (=Initialwert)

Parameter der Ausgänge

12673

Parameter	Datentyp	Beschreibung
READY	BOOL	TRUE: Funktionsbaustein ist aktiv Die Werte an den FB-Ausgängen sind gültig FALSE (= Initialwert): Funktionsbaustein ist nicht aktiv
S_ENABLESWITCH_OUT	BOOL	Sicherheitsausgang TRUE: Sicherheitsfunktion freigegeben FALSE (= Initialwert): Sicherheitsfunktion gesperrt
SAFETYDEMAND	BOOL	TRUE: Sicherheitsfunktion ist angefordert Aktion des Maschinenführers erforderlich FALSE (= Initialwert): Sicherheitsfunktion nicht angefordert
RESETREQUEST	BOOL	Ein Reset ist erforderlich, damit der FB weiterarbeiten kann TRUE: Reset ist erforderlich FALSE (= Initialwert): kein Reset erforderlich
ERROR	BOOL	TRUE: ein Fehler ist aufgetreten FB ist im Fehlerzustand FALSE (= Initialwert): kein Fehler aufgetreten
DIAGCODE	WORD	Informationen zum aktuellen Zustand des FBs (mögliche Meldungen → folgende Tabelle)

Mögliche Ergebnisse für DIAGCODE:

Wert [hex]	Beschreibung
0000	(Initialwert) Funktionsbaustein ist nicht aktiv
8000	Sicherheitsausgang ist freigegeben
8010	Funktionsbaustein ist bereit + Sicherheitsmodus ist aktiv
8802	Funktionsbaustein ist bereit + Sicherheitsmodus ist angefordert
8812	Funktionsbaustein ist bereit + Sicherheitsmodus ist aktiv FB wartet auf Sicherheitseingang
C001	im Zustand C410 oder C810 ist RESET-Signal statisch FB wartet auf RESET=FALSE
C410	Fehler im Ablauf bei Aktivierung von S_SAFETYACTIVE: Sicherheitseingang war bereits TRUE
C810	Fehler im Ablauf bei Aktivierung von S_SAFETYACTIVE: Sicherheitseingang war bereits TRUE FB wartet auf RESET

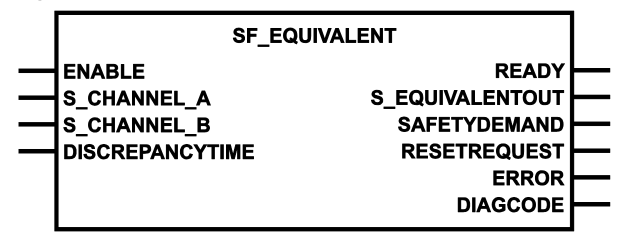
SF_EQUIVALENT

12393

Baustein-Typ = Funktionsbaustein (FB)

Baustein ist enthalten in Bibliothek `ifm_SafetyPLCopen_Vxxyyzz.lib`

Symbol in CODESYS:



Beschreibung

12395

SF_EQUIVALENT vergleicht zwei binäre sichere Eingänge (A und B) miteinander.
Der FB überwacht den Zeitabstand zwischen den Flankenwechseln auf beiden Eingängen.

15749

> ❗ Sind die Eingänge des FB mit einem OSSD-Sicherheitsschalter belegt, dann den FB-Eingang DISCREPANCYTIME mit einem Wert ≥ 10 ms einstellen (je nach Applikation).
> Andernfalls kann der FB fälschlich eine Fehlstellung der Eingänge melden.

Parameter der Eingänge

12396

Parameter	Datentyp	Beschreibung
ENABLE	BOOL	TRUE: Baustein ausführen FALSE: Baustein wird nicht ausgeführt > Baustein-Eingänge sind nicht aktiv > Baustein-Ausgänge sind nicht spezifiziert
S_CHANNEL_A	BOOL	binärer sicherheitsrelevanter Eingang A TRUE: Kontakt geschlossen FALSE (= Initialwert): Kontakt offen
S_CHANNEL_B	BOOL	binärer sicherheitsrelevanter Eingang B TRUE: Kontakt geschlossen FALSE (= Initialwert): Kontakt offen
DISCREPANCYTIME	TIME	Maximal zulässige Zeitdifferenz zwischen den Flankenwechseln an den Eingängen A und B von FALSE nach TRUE oder von TRUE nach FALSE. Die Überwachung der Zeitdifferenz ist abhängig von der SPS-Zykluszeit. Initialwert = T#0ms

Parameter der Ausgänge

12401

Parameter	Datentyp	Beschreibung
READY	BOOL	TRUE: Funktionsbaustein ist aktiv Die Werte an den FB-Ausgängen sind gültig FALSE (= Initialwert): Funktionsbaustein ist nicht aktiv
S_EQUIVALENTOUT	BOOL	Sicherheitsausgang TRUE: Beide Eingänge (A/B) = TRUE **und** die letzten Flankenwechsel auf den Eingängen waren innerhalb der Überwachungszeit FALSE (= Initialwert): Mindestens ein Eingang (A/B) = FALSE **oder** ein Flankenwechsel war außerhalb der Überwachungszeit
SAFETYDEMAND	BOOL	TRUE: Sicherheitsfunktion ist angefordert Aktion des Maschinenführers erforderlich FALSE (= Initialwert): Sicherheitsfunktion nicht angefordert
RESETREQUEST	BOOL	(FB-Ausgang hier ohne Funktion)
ERROR	BOOL	TRUE: ein Fehler ist aufgetreten FB ist im Fehlerzustand FALSE (= Initialwert): kein Fehler aufgetreten
DIAGCODE	WORD	Informationen zum aktuellen Zustand des FBs (mögliche Meldungen → folgende Tabelle)

Mögliche Ergebnisse für DIAGCODE:

Wert [hex]	Beschreibung
0000	(Initialwert) Funktionsbaustein ist nicht aktiv
8000	Sicherheitsausgang ist freigegeben
8001	Funktionsbaustein ist bereit: READY=TRUE kein Sicherheitseingang ist aktiv
8004	Sicherheitseingang 1 ist aktiv Sicherheitseingang 2 ist noch nicht aktiv Überwachungszeit läuft
8005	einer der beiden Sicherheitseingänge ist noch aktiv der andere Sicherheitseingang ist nicht mehr aktiv Überwachungszeit läuft
8014	Sicherheitseingang 1 ist noch nicht aktiv Sicherheitseingang 2 ist aktiv Überwachungszeit läuft
C001	Überwachungszeit abgelaufen (FB wartet auf Kanal 2)
C002	Überwachungszeit abgelaufen (FB wartet auf Kanal 1)
C003	Überwachungszeit abgelaufen (FB wartet auf beide Kanäle=inaktiv)

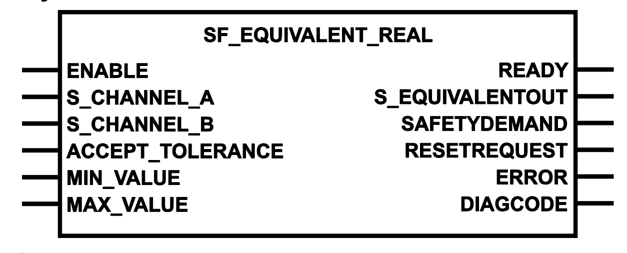
SF_EQUIVALENT_REAL

12463

Baustein-Typ = Funktionsbaustein (FB)

Baustein ist enthalten in Bibliothek `ifm_SafetyPLCopen_Vxxyyzz.lib`

Symbol in CODESYS:



Beschreibung

12465

SF_EQUIVALENT_REAL vergleicht zwei sichere Eingangswerte (A und B) vom Typ REAL miteinander.
Der FB überwacht die Werte auf Abweichung zueinander.
Der FB überwacht die Werte auf zulässige Minimal- und Maximalwerte.

Parameter der Eingänge

12466

Parameter	Datentyp	Beschreibung
ENABLE	BOOL	TRUE: Baustein ausführen FALSE: Baustein wird nicht ausgeführt > Baustein-Eingänge sind nicht aktiv > Baustein-Ausgänge sind nicht spezifiziert
S_CHANNEL_A	REAL	Eingang A zum Einlesen eines Wertes vom Typ REAL (z. B. Frequenzwert) Initialwert = 0.0
S_CHANNEL_B	REAL	Eingang B zum Einlesen eines Wertes vom Typ REAL (z. B. Frequenzwert) Initialwert = 0.0
ACCEPT_TOLERANCE	BYTE	zulässige Abweichung (±) von S_CHANNEL_B in [%] bezogen auf S_CHANNEL_A Initialwert = 0x00
MIN_VALUE	REAL	Mindestwert an den Eingängen A und B Initialwert = 0.0
MAX_VALUE	REAL	Maximalwert an den Eingängen A und B Initialwert = 0.0

Parameter der Ausgänge

12467

Parameter	Datentyp	Beschreibung
READY	BOOL	TRUE: Funktionsbaustein ist aktiv Die Werte an den FB-Ausgängen sind gültig FALSE (= Initialwert): Funktionsbaustein ist nicht aktiv
S_EQUIVALENTOUT	BOOL	Sicherheitsausgang TRUE: Werte beider Eingänge im zulässigen Wertebereich **und** die Differenz zwischen den Eingangswerten ist kleiner/gleich der zulässigen Differenz FALSE (= Initialwert): Werte des Eingangs A und/oder B sind außerhalb des zulässigen Wertebereichs und/oder die Differenz zwischen den Eingangswerten ist größer als zulässig
SAFETYDEMAND	BOOL	TRUE: Sicherheitsfunktion ist angefordert Aktion des Maschinenführers erforderlich FALSE (= Initialwert): Sicherheitsfunktion nicht angefordert
RESETREQUEST	BOOL	(FB-Ausgang hier ohne Funktion)
ERROR	BOOL	TRUE: ein Fehler ist aufgetreten FB ist im Fehlerzustand FALSE (= Initialwert): kein Fehler aufgetreten
DIAGCODE	WORD	Informationen zum aktuellen Zustand des FBs (mögliche Meldungen → folgende Tabelle)

Mögliche Ergebnisse für DIAGCODE:

Wert [hex]	Beschreibung
0000	(Initialwert) Funktionsbaustein ist nicht aktiv
8000	Sicherheitsausgang ist freigegeben
8001	Funktionsbaustein ist bereit: READY=TRUE kein Sicherheitseingang ist aktiv
C001	Parametrierfehler: Toleranz > 100 %
C002	Parametrierfehler: Min-Wert > Max-Wert
C003	Eingangswert von Kanal A > Max-Wert
C004	Eingangswert von Kanal A < Min-Wert
C005	Eingangswert von Kanal B > Max-Wert
C006	Eingangswert von Kanal B < Min-Wert
C007	Eingangswert von Kanal B > (Wert von Kanal A + Toleranz)
C008	Eingangswert von Kanal B < (Wert von Kanal A - Toleranz)

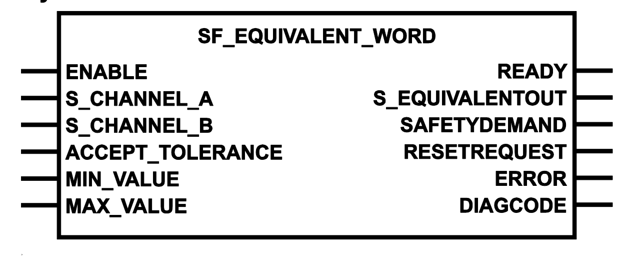
SF_EQUIVALENT_WORD

12425

Baustein-Typ = Funktionsbaustein (FB)

Baustein ist enthalten in Bibliothek `ifm_SafetyPLCopen_Vxxyyzz.lib`

Symbol in CODESYS:



Beschreibung

12427

SF_EQUIVALENT_WORD vergleicht zwei sichere Eingangswerte (A und B) vom Typ WORD miteinander.
Der FB überwacht die Werte auf Abweichung zueinander.
Der FB überwacht die Werte auf zulässige Minimal- und Maximalwerte.

Parameter der Eingänge

12432

Parameter	Datentyp	Beschreibung
ENABLE	BOOL	TRUE: Baustein ausführen FALSE: Baustein wird nicht ausgeführt > Baustein-Eingänge sind nicht aktiv > Baustein-Ausgänge sind nicht spezifiziert
S_CHANNEL_A	WORD	Eingang A zum Einlesen eines Wertes vom Typ WORD (z. B. Analogwert) Initialwert = 0x0000
S_CHANNEL_B	WORD	Eingang B zum Einlesen eines Wertes vom Typ WORD (z. B. Analogwert) Initialwert = 0x0000
ACCEPT_TOLERANCE	BYTE	zulässige Abweichung (±) von S_CHANNEL_B in [%] bezogen auf S_CHANNEL_A Initialwert = 0x00
MIN_VALUE	WORD	Mindestwert an den Eingängen A und B Initialwert = 0x0000
MAX_VALUE	WORD	Maximalwert an den Eingängen A und B Initialwert = 0x0000

Parameter der Ausgänge

12434

Parameter	Datentyp	Beschreibung
READY	BOOL	TRUE: Funktionsbaustein ist aktiv Die Werte an den FB-Ausgängen sind gültig FALSE (= Initialwert): Funktionsbaustein ist nicht aktiv
S_EQUIVALENTOUT	BOOL	Sicherheitsausgang TRUE: Werte beider Eingänge im zulässigen Wertebereich **und** die Differenz zwischen den Eingangswerten ist kleiner/gleich der zulässigen Differenz FALSE (= Initialwert): Werte des Eingangs A und/oder B sind außerhalb des zulässigen Wertebereichs und/oder die Differenz zwischen den Eingangswerten ist größer als zulässig
SAFETYDEMAND	BOOL	TRUE: Sicherheitsfunktion ist angefordert Aktion des Maschinenführers erforderlich FALSE (= Initialwert): Sicherheitsfunktion nicht angefordert
RESETREQUEST	BOOL	(FB-Ausgang hier ohne Funktion)
ERROR	BOOL	TRUE: ein Fehler ist aufgetreten FB ist im Fehlerzustand FALSE (= Initialwert): kein Fehler aufgetreten
DIAGCODE	WORD	Informationen zum aktuellen Zustand des FBs (mögliche Meldungen → folgende Tabelle)

Mögliche Ergebnisse für DIAGCODE:

Wert [hex]	Beschreibung
0000	(Initialwert) Funktionsbaustein ist nicht aktiv
8000	Sicherheitsausgang ist freigegeben
8001	Funktionsbaustein ist bereit: READY=TRUE kein Sicherheitseingang ist aktiv
C001	Parametrierfehler: Toleranz > 100 %
C002	Parametrierfehler: Min-Wert > Max-Wert
C003	Eingangswert von Kanal A > Max-Wert
C004	Eingangswert von Kanal A < Min-Wert
C005	Eingangswert von Kanal B > Max-Wert
C006	Eingangswert von Kanal B < Min-Wert
C007	Eingangswert von Kanal B > (Wert von Kanal A + Toleranz)
C008	Eingangswert von Kanal B < (Wert von Kanal A - Toleranz)

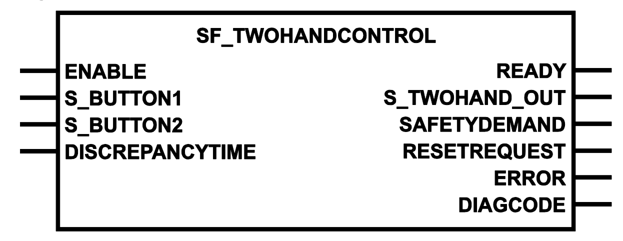
SF_TWOHANDCONTROL

12530

Baustein-Typ = Funktionsbaustein (FB)

Baustein ist enthalten in Bibliothek `ifm_SafetyPLCopen_Vxxyyzz.lib`

Symbol in CODESYS:



Beschreibung

12533

Der FB SF_TWOHANDCONTROL dient zum Realisieren einer Zweihandbedienung.
Der FB überwacht den Zeitabstand zwischen dem Aktivieren der beiden Taster.

Parameter der Eingänge

12534

Parameter	Datentyp	Beschreibung
ENABLE	BOOL	TRUE: Baustein ausführen FALSE: Baustein wird nicht ausgeführt > Baustein-Eingänge sind nicht aktiv > Baustein-Ausgänge sind nicht spezifiziert
S_BUTTON1	BOOL	Sicherheitstaster 1 TRUE: Taster betätigt: Kontakt geschlossen FALSE (= Initialwert): Taster losgelassen: Kontakt offen
S_BUTTON2	BOOL	Sicherheitstaster 2 TRUE: Taster betätigt: Kontakt geschlossen FALSE (= Initialwert): Taster losgelassen: Kontakt offen
DISCREPANCYTIME	TIME	Maximal zulässige Zeitdifferenz zwischen dem Betätigen der zwei Taster. Initialwert = T#0ms Maximalwert = T#500ms

Parameter der Ausgänge

12535

Parameter	Datentyp	Beschreibung
READY	BOOL	TRUE: Funktionsbaustein ist aktiv Die Werte an den FB-Ausgängen sind gültig FALSE (= Initialwert): Funktionsbaustein ist nicht aktiv
S_TWOHAND_OUT	BOOL	TRUE: Beide Taster werden betätigt Wechsel FALSE ⇨ TRUE erfolgte innerhalb der zulässigen Zeit FALSE (= Initialwert): keine Zweihandbedienung ausgeführt Zweihandbedienung fehlerhaft ausgeführt
SAFETYDEMAND	BOOL	TRUE: Sicherheitsfunktion ist angefordert Aktion des Maschinenführers erforderlich FALSE (= Initialwert): Sicherheitsfunktion nicht angefordert
RESETREQUEST	BOOL	(FB-Ausgang hier ohne Funktion)
ERROR	BOOL	TRUE: ein Fehler ist aufgetreten FB ist im Fehlerzustand FALSE (= Initialwert): kein Fehler aufgetreten
DIAGCODE	WORD	Informationen zum aktuellen Zustand des FBs (mögliche Meldungen → folgende Tabelle)

Mögliche Ergebnisse für DIAGCODE:

Wert [hex]	Beschreibung
0000	(Initialwert) Funktionsbaustein ist nicht aktiv
8000	Sicherheitsausgang ist freigegeben
8001	Funktionsbaustein ist bereit: READY=TRUE kein Sicherheitseingang ist aktiv
8004	Funktionsbaustein ist bereit: READY=TRUE FB wartet auf Sicherheitseingänge=TRUE
8005	Sicherheitseingang 1 ist aktiv Sicherheitseingang 2 ist noch nicht aktiv Überwachungszeit läuft
8006	Sicherheitseingang 1 ist noch nicht aktiv Sicherheitseingang 2 ist aktiv Überwachungszeit läuft
8007	Sicherheitsausgang wurde FALSE Sicherheitseingang 1 ist noch aktiv Sicherheitseingang 2 ist nicht mehr aktiv
8008	Sicherheitsausgang wurde FALSE Sicherheitseingang 1 ist nicht mehr aktiv Sicherheitseingang 2 ist noch aktiv
8009	Sicherheitsausgang wurde FALSE Sicherheitseingang 1 ist noch oder wieder aktiv Sicherheitseingang 2 ist noch oder wieder aktiv FB wartet auf beide Eingänge=FALSE
8019	Ablauf der Sicherheitseingänge nicht korrekt FB wartet auf beide Eingänge=FALSE
C001	Fehler im Ablauf bei FB-Start: Sicherheitseingang 1 war bereits TRUE
C002	Fehler im Ablauf bei FB-Start: Sicherheitseingang 2 war bereits TRUE

Wert [hex]	Beschreibung
C003	Fehler im Ablauf bei FB-Start: beide Sicherheitseingänge waren bereits TRUE
C004	Überwachungszeit abgelaufen (FB wartet auf Kanal 1)
C005	Überwachungszeit abgelaufen (FB wartet auf Kanal 2)
C006	Überwachungszeit abgelaufen, aber beide Kanäle=TRUE Dieser Zustand ist nur möglich, wenn... • beide Eingänge wechseln von divergent nach konvergent (=TRUE) • gleichzeitig im selben Zyklus Überwachungszeit abgelaufen

6.2.11 Bausteine: analoge Werte anpassen

Inhalt

1603

Wenn die Werte analoger Eingänge oder die Ergebnisse von analogen Funktionen angepasst werden müssen, helfen Ihnen die folgenden Funktionsbausteine.

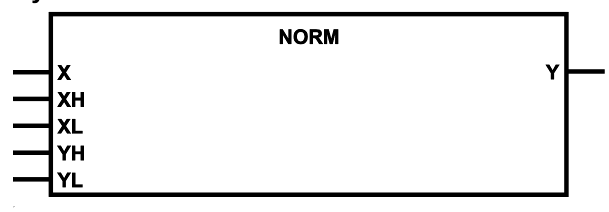
NORM

401

Baustein-Typ = Funktionsbaustein (FB)

Baustein ist enthalten in Bibliothek `ifm_CR7132_Vxxyyzz.LIB`

Symbol in CODESYS:



Beschreibung

404

NORM normiert einen Wert innerhalb festgelegter Grenzen auf einen Wert mit neuen Grenzen.

Der FB normiert einen Wert vom Typ WORD, der innerhalb der Grenzen XH und XL liegt, auf einen Ausgangswert innerhalb der Grenzen YH und YL. Der FB wird z.B. bei der Erzeugung von PWM-Werten aus analogen Eingangsgrößen genutzt.

HINWEIS

► Der Eingangswert für X muss sich im definierten Bereich zwischen XL und XH befinden!
Der FB prüft NICHT den Wert X auf Plausibilität.

> Bedingt durch die Rundungsfehler können Abweichungen beim normierten Wert um 1 auftreten.

> Werden die Grenzen (XH/XL oder YH/YL) invertiert angegeben, erfolgt auch die Normierung invertiert.

Parameter der Eingänge

405

Parameter	Datentyp	Beschreibung
X	WORD	Eingangswert
XH	WORD	obere Grenze des Eingangswertebereichs [Inkremente]
XL	WORD	untere Grenze des Eingangswertebereichs [Inkremente]
YH	WORD	obere Grenze des Ausgangswertebereichs
YL	WORD	untere Grenze des Ausgangswertebereichs

Parameter der Ausgänge

406

Parameter	Datentyp	Beschreibung
Y	WORD	Ausgangswert

Beispiel 1

407

unterer Grenzwert Eingang	0	XL
oberer Grenzwert Eingang	100	XH
unterer Grenzwert Ausgang	0	YL
oberer Grenzwert Ausgang	2000	YH

dann wandelt der Funktionsbaustein das Eingangssignal z.B. wie folgt um:

von X =	50	0	100	75
	↓	↓	↓	↓
nach Y =	1000	0	2000	1500

Beispiel 2

408

unterer Grenzwert Eingang	2000	XL
oberer Grenzwert Eingang	0	XH
unterer Grenzwert Ausgang	0	YL
oberer Grenzwert Ausgang	100	YH

dann wandelt der Funktionsbaustein das Eingangssignal z.B. wie folgt um:

von X =	1000	0	2000	1500
	↓	↓	↓	↓
nach Y =	50	100	0	25

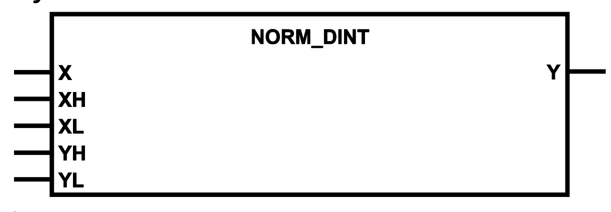
NORM_DINT

2217

Baustein-Typ = Funktionsbaustein (FB)

Baustein ist enthalten in Bibliothek `ifm_CR7132_Vxxyyzz.LIB`

Symbol in CODESYS:



Beschreibung

2355

NORM_DINT normiert einen Wert innerhalb festgelegter Grenzen auf einen Wert mit neuen Grenzen.

Der FB normiert einen Wert vom Typ DINT, der innerhalb der Grenzen XH und XL liegt, auf einen Ausgangswert innerhalb der Grenzen YH und YL. Dieser FB wird z.B. bei der Erzeugung von PWM-Werten aus analogen Eingangsgrößen genutzt.

> **HINWEIS**
>
> ► Der Eingangswert für X muss sich im definierten Bereich zwischen XL und XH befinden!
> Der FB prüft NICHT den Wert X auf Plausibilität.
>
> ► Das Ergebnis der Berechnung (XH-XL)•(YH-YL) muss im Wertebereich des Datentyps DINT (-2 147 483 648...2 147 483 647) bleiben!
>
> > Bedingt durch die Rundungsfehler können Abweichungen beim normierten Wert um 1 auftreten.
>
> > Werden die Grenzen (XH/XL oder YH/YL) invertiert angegeben, erfolgt auch die Normierung invertiert.

Parameter der Eingänge

2359

Parameter	Datentyp	Beschreibung
X	DINT	Eingangswert
XH	DINT	obere Grenze des Eingangswertebereichs
XL	DINT	untere Grenze des Eingangswertebereichs
YH	DINT	obere Grenze des Ausgangswertebereichs
YL	DINT	untere Grenze des Ausgangswertebereichs

Parameter der Ausgänge

2360

Parameter	Datentyp	Beschreibung
Y	DINT	Ausgangswert

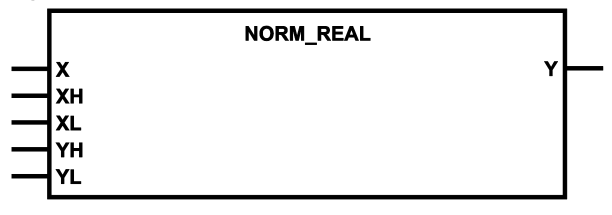
NORM_REAL

2218

Baustein-Typ = Funktionsbaustein (FB)

Baustein ist enthalten in Bibliothek `ifm_CR7132_Vxxyyzz.LIB`

Symbol in CODESYS:



Beschreibung

2358

NORM_REAL normiert einen Wert innerhalb festgelegter Grenzen auf einen Wert mit neuen Grenzen.

Der FB normiert einen Wert vom Typ REAL, der innerhalb der Grenzen XH und XL liegt, auf einen Ausgangswert innerhalb der Grenzen YH und YL. Dieser FB wird z.B. bei der Erzeugung von PWM-Werten aus analogen Eingangsgrößen genutzt.

HINWEIS

► Der Eingangswert für X muss sich im definierten Bereich zwischen XL und XH befinden!
Der FB prüft NICHT den Wert X auf Plausibilität.

► Das Ergebnis der Berechnung (XH-XL)•(YH-YL) muss im Wertebereich des Datentyps REAL (-3,402823466•10^{38}...3,402823466•10^{38}) bleiben!

> Bedingt durch die Rundungsfehler können Abweichungen beim normierten Wert um 1 auftreten.

> Werden die Grenzen (XH/XL oder YH/YL) invertiert angegeben, erfolgt auch die Normierung invertiert.

Parameter der Eingänge

2356

Parameter	Datentyp	Beschreibung
X	REAL	Eingangswert
XH	REAL	obere Grenze des Eingangswertebereichs
XL	REAL	untere Grenze des Eingangswertebereichs
YH	REAL	obere Grenze des Ausgangswertebereichs
YL	REAL	untere Grenze des Ausgangswertebereichs

Parameter der Ausgänge

2357

Parameter	Datentyp	Beschreibung
Y	REAL	Ausgangswert

6.2.12 Bausteine: Zählerfunktionen zur Frequenz- und Periodendauermessung

Inhalt

2322

Je nach Controller werden bis zu 16*) schnelle Eingänge unterstützt, die Eingangsfrequenzen bis zu 30 kHz verarbeiten können. Neben der reinen Frequenzmessung an den Eingängen FRQ können die Eingänge ENC auch zur Auswertung von inkrementellen Drehgebern (Zählerfunktion) eingesetzt werden.
*) ExtendedController: bis zu 32 schnelle Eingänge

Bedingt durch die unterschiedlichen Messmethoden können Fehler bei der Frequenzermittlung auftreten.

Zur einfachen Auswertung stehen folgende Bausteine zur Verfügung:

Baustein	zulässige Werte	Erklärung
FREQUENCY	0,1...30 000 Hz	Frequenz am angegebenen Kanal messen. Messfehler verringert sich bei hohen Frequenzen
PERIOD	0,1...5 000 Hz	Frequenz und Periodendauer (Zykluszeit) am angegebenen Kanal messen
PERIOD_RATIO	0,1...5 000 Hz	Frequenz und Periodendauer (Zykluszeit) sowie Puls-Pause-Verhältnis [‰] am angegebenen Kanal messen
FREQUENCY_PERIOD	0,1...30 000 Hz	Die Funktion vereinigt die beiden Funktionen FREQUENCY und PERIOD oder PERIOD_RATIO. Automatisches Umschalten der Messmethode bei 5 kHz
PHASE	0,1...5 000 Hz	Liest ein Kanalpaar ein und vergleicht die Phasenlage der Signale
INC_ENCODER	0,1...30 000 Hz	Vorwärts-/Rückwärts-Zählerfunktion zur Auswertung von Drehgebern
FAST_COUNT	0,1...50 000 Hz	Schnelle Impulse zählen

Wichtig bei Einsatz der schnellen Eingänge als "normale" Digitaleingänge:

- ► Die erhöhte Empfindlichkeit gegen Störimpulse beachten (z.B. Kontaktprellen bei mechanischen Kontakten).
- ► Das Eingangssignal bei Bedarf entprellen! → Kapitel ***Hardware-Filter konfigurieren*** (→ Seite 182)
- Der Standard-Digitaleingang kann Signale bis 50 Hz auswerten.

FAST_COUNT

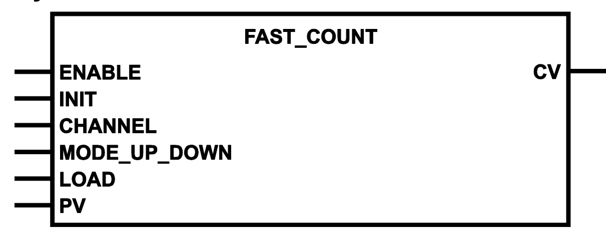
567

Baustein-Typ = Funktionsbaustein (FB)

Baustein ist enthalten in Bibliothek `ifm_CR7132_Vxxyyzz.LIB`

Für die Extended-Seite des ExtendedControllers endet der FB-Name mit "_E".

Symbol in CODESYS:



Beschreibung

6830

FAST_COUNT arbeitet als Zählerbaustein für schnelle Eingangsimpulse.

Während ENABLE=TRUE erfasst der FB steigende Flanken an den FRQ-Eingangskanälen.
Maximale Eingangsfrequenz → Datenblatt.

Nach Rücksetzen und erneutem Setzen von ENABLE zählt der Zähler von dem Wert an weiter, der beim letzten Rücksetzen von ENABLE gültig war.

Mit Setzen von INIT (steigende Flanke) wird der Zählerwert CV=0 gesetzt.
Nach Rücksetzen des Parameters INIT zählt der Zähler vom angegebenen Startwert an.

Am selben Eingang diesen FB **nicht** gemeinsam mit einem der folgenden FBs nutzen!
- *FREQUENCY* (→ Seite 300)
- *FREQUENCY_PERIOD* (→ Seite 302)
- *INC_ENCODER* (→ Seite 304)
- *PERIOD* (→ Seite 306)
- *PERIOD_RATIO* (→ Seite 308)
- *PHASE* (→ Seite 310)

Parameter der Eingänge

571

Parameter	Datentyp	Beschreibung
ENABLE	BOOL	TRUE: Baustein ausführen FALSE: Baustein wird nicht ausgeführt > Zähler angehalten
INIT	BOOL	FALSE ⇨ TRUE (Flanke): Baustein wird initialisiert FALSE: im weiteren Programmablauf
CHANNEL	BYTE	Nummer des schnellen Eingangskanals (0...15) 0...15 für die Eingänge I00...I15 Für den FB xxx_**E** (falls vorhanden) gilt: 0...15 für die Eingänge I00_E...I15_E
MODE_UP_DOWN	BOOL	TRUE: Zähler zählt abwärts FALSE: Zähler zählt aufwärts
LOAD	BOOL	TRUE: Startwert PV wird in CV geladen FALSE: Funktion wird nicht ausgeführt
PV	DWORD	Startwert (Preset value) für den Zähler

Parameter der Ausgänge

572

Parameter	Datentyp	Beschreibung
CV	DWORD	aktueller Zählerwert

FREQUENCY

12903

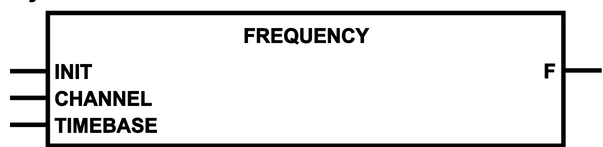
Baustein-Typ = Funktionsbaustein (FB)

Baustein ist enthalten in Bibliothek `ifm_CR7132_Vxxyyzz.LIB`

> Für Sicherheitssignale die Messwerte →diversitär auswerten!
> Zusätzlich zu diesem FB den FB ***SF_EQUIVALENT_REAL*** (→ Seite 285) einsetzen!

Für die Extended-Seite des ExtendedControllers endet der FB-Name mit "_E".

Symbol in CODESYS:



Beschreibung

2325

FREQUENCY misst die Frequenz des am gewählten Kanal (CHANNEL) ankommenden Signals. Der FB wertet dazu die positive Flanke des Signals aus. In Abhängigkeit von der Zeitbasis (TIMEBASE) können Frequenzmessungen in einem weiten Wertebereich durchgeführt werden. Hohe Frequenzen erfordern eine kurze Zeitbasis, niedrige eine entsprechend längere. Die Frequenz wird direkt in [Hz] ausgegeben.

> Sicherstellen, dass der FB innerhalb des Wertes von TIMEBASE nicht mehr als 65 535 positive Flanken empfängt!
> Sonst kann das interne Zählregister überlaufen und zu falschen Ergebnissen führen.

> Am selben Eingang diesen FB **nicht** gemeinsam mit einem der folgenden FBs nutzen!
> - ***FAST_COUNT*** (→ Seite 298)
> - ***FREQUENCY_PERIOD*** (→ Seite 302)
> - ***INC_ENCODER*** (→ Seite 304)
> - ***PERIOD*** (→ Seite 306)
> - ***PERIOD_RATIO*** (→ Seite 308)
> - ***PHASE*** (→ Seite 310)

Parameter der Eingänge

2599

Parameter	Datentyp	Beschreibung
INIT	BOOL	TRUE (nur 1 Zyklus lang): Baustein und Schnittstelle werden initialisiert FALSE: Messung läuft oder: Messung startet, wenn zuvor INIT=TRUE war
CHANNEL	BYTE	Nummer des schnellen Eingangskanals (0...15) 0...15 für die Eingänge I00...I15 Für den FB xxx**_E** (falls vorhanden) gilt: 0...15 für die Eingänge I00_E...I15_E
TIMEBASE	TIME	Zeitbasis zur Frequenzmessung (max. 57 s)

Parameter der Ausgänge

542

Parameter	Datentyp	Beschreibung
F	REAL	Frequenz des Eingangssignals in [Hz]

FREQUENCY_PERIOD

12904

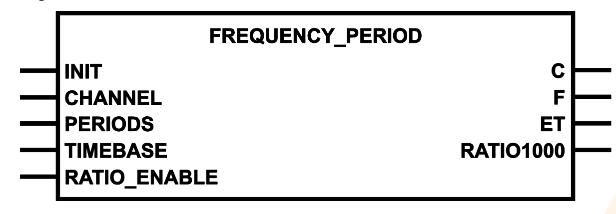
Baustein-Typ = Funktionsbaustein (FB)

Baustein ist enthalten in Bibliothek `ifm_CR7132_Vxxyyzz.LIB`

> Für Sicherheitssignale die Messwerte →diversitär auswerten!
> Zusätzlich zu diesem FB den FB ***SF_EQUIVALENT_REAL*** (→ Seite 285) einsetzen!

Für die Extended-Seite des ExtendedControllers endet der FB-Name mit "_E".

Symbol in CODESYS:



Beschreibung

2335

FREQUENCY_PERIOD misst die Frequenz und die Periodendauer (Zykluszeit) in [µs] am angegebenen Kanal (für alle Eingänge zugelassen). Maximale Eingangsfrequenz → Datenblatt.

Der FB vereinigt PERIOD oder PERIOD_RATIO und FREQUENCY in einem gemeinsamen Funktionsbaustein. Die Umschaltung der Messmethode erfolgt automatisch bei 5 kHz:

- unterhalb von 5 kHz verhält sich der FB wie PERIOD oder PERIOD_RATIO
- oberhalb von 5 kHz verhält sich der FB wie FREQUENCY.

Der FB misst die Frequenz und die Zykluszeit des am gewählten Kanal (CHANNEL) anstehenden Signals. Zur Berechnung werden alle positiven Flanken ausgewertet und der Mittelwert über die Anzahl der angegebenen Perioden (PERIODS) gebildet.

Bei einer Eingangsfrequenz > 5 kHz und aktivem FREQUENCY-Modus kann der Ratio nicht gemessen werden.

Der maximale Messbereich beträgt ca. 15 min.

> Sicherstellen, dass der FB innerhalb des Wertes von TIMEBASE nicht mehr als 65 535 positive Flanken empfängt!
> Sonst kann das interne Zählregister überlaufen und zu falschen Ergebnissen führen.

HINWEIS

► Am selben Eingang diesen FB **nicht** gemeinsam mit einem der folgenden FBs nutzen!

- ***FAST_COUNT*** (→ Seite 298)
- ***FREQUENCY*** (→ Seite 300)
- ***INC_ENCODER*** (→ Seite 304)
- ***PERIOD*** (→ Seite 306)
- ***PERIOD_RATIO*** (→ Seite 308)
- ***PHASE*** (→ Seite 310)

Frequenzen < 5 Hz werden nicht mehr sicher angegeben.

Parameter der Eingänge

2336

Parameter	Datentyp	Beschreibung
INIT	BOOL	TRUE (nur 1 Zyklus lang): Baustein und Schnittstelle werden initialisiert FALSE: Messung läuft oder: Messung startet, wenn zuvor INIT=TRUE war
CHANNEL	BYTE	Nummer des schnellen Eingangskanals (0...15) 0...15 für die Eingänge I00...I15 Für den FB xxx**_E** (falls vorhanden) gilt: 0...15 für die Eingänge I00_E...I15_E
PERIODS	BYTE	Anzahl der Perioden, über die gemittelt wird (1...16) 0 : Ausgänge C und F werden nicht aktualisiert > 16 : wird auf 16 limitiert
TIMEBASE	TIME	Zeitbasis zur Frequenzmessung (max. 57 s)
RATIO_ENABLE	BOOL	TRUE: Ratio-Messung an RATIO1000 ausgeben FALSE: Ratio-Messung nicht ausgeben

Parameter der Ausgänge

14315

Parameter	Datentyp	Beschreibung
C	DWORD	Zykluszeit der erfassten Perioden in [µs] zulässig = 33...10 000 000 = 0x21...0x989680
F	REAL	Frequenz des Eingangssignals in [Hz]
ET	TIME	RATIO_ENABLE = TRUE: Verstrichene Zeit seit dem letzten Flankenwechsel am Eingang RATIO_ENABLE = FALSE: Verstrichene Zeit seit der letzten positiven Flanke am Eingang (nutzbar bei sehr langsamen Signalen)
RATIO1000	WORD	Puls-/Periode-Verhältnis in [‰] Voraussetzungen: • Periodendauermessung • Impulsdauer ≥ 100 µs • Frequenz < 5 kHz Wert nicht für sicherheitsrelevante Funktionen verwenden!

INC_ENCODER

525

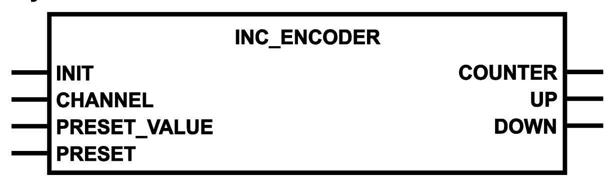
= Incremental Encoder

Baustein-Typ = Funktionsbaustein (FB)

Baustein ist enthalten in Bibliothek `ifm_CR7132_Vxxyyzz.LIB`

Für die Extended-Seite des ExtendedControllers endet der FB-Name mit "_E".

Symbol in CODESYS:



Beschreibung

2602

INC_ENCODER bietet eine Vorwärts-/Rückwärts-Zählerfunktion zur Auswertung von Drehgebern.

Immer zwei Frequenzeingänge bilden das Eingangspaar, das über den FB ausgewertet wird.

Grenzfrequenz = 30 kHz
max. anschließbar: 4 Drehgeber (ExtendedController: max. 8 Drehgeber)

Voreinstellwert setzen:
1. Wert in PRESET_VALUE eintragen
2. PRESET für einen Zyklus auf TRUE setzen
3. PRESET wieder auf FALSE setzen

Der FB zählt die Impulse an den Eingängen, solange INIT=FALSE und PRESET=FALSE sind.
Am Ausgang COUNTER steht der aktuelle Zählerstand an.

Die Ausgänge UP und DOWN zeigen die aktuelle Zählrichtung des Zählers an. Die Ausgänge sind dann TRUE, wenn im vorangegangenen Programmzyklus der Zähler in die entsprechende Richtung gezählt hat. Bleibt der Zähler stehen, wird auch der Richtungsausgang im folgenden Programmzyklus zurückgesetzt.

Am selben Eingang diesen FB **nicht** gemeinsam mit einem der folgenden FBs nutzen!
- *FAST_COUNT* (→ Seite 298)
- *FREQUENCY* (→ Seite 300)
- *FREQUENCY_PERIOD* (→ Seite 302)
- *PERIOD* (→ Seite 306)
- *PERIOD_RATIO* (→ Seite 308)
- *PHASE* (→ Seite 310)

Parameter der Eingänge

529

Parameter	Datentyp	Beschreibung
INIT	BOOL	TRUE (nur 1 Zyklus lang): Baustein wird initialisiert FALSE: im weiteren Programmablauf
CHANNEL	BYTE	Nummer des Eingangskanal-Paares (0...3) 0 = Kanalpaar 0 = Eingänge I00 + I01 ... 3 = Kanalpaar 3 = Eingänge I06 + I07 Für den FB xxx_**E** (falls vorhanden) gilt: 0 = Kanalpaar 0 = Eingänge I00_E + I01_E ... 3 = Kanalpaar 3 = Eingänge I06_E + I07_E
PRESET_VALUE	DINT	Zähler-Startwert
PRESET	BOOL	FALSE ⇨ TRUE (Flanke): PRESET_VALUE wird nach COUNTER geladen TRUE: Zähler ignoriert die Eingangsimpulse FALSE: Zähler zählt die Eingangsimpulse

Parameter der Ausgänge

530

Parameter	Datentyp	Beschreibung
COUNTER	DINT	aktueller Zählerstand
UP	BOOL	TRUE: Zähler zählt aufwärts FALSE: Zähler steht
DOWN	BOOL	TRUE: Zähler zählt abwärts FALSE: Zähler steht

PERIOD

11868

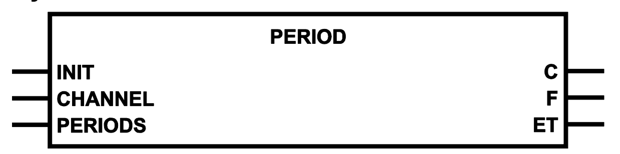
Baustein-Typ = Funktionsbaustein (FB)

Baustein ist enthalten in Bibliothek `ifm_CR7132_Vxxyyzz.LIB`

> Für Sicherheitssignale die Messwerte →diversitär auswerten!
> Zusätzlich zu diesem FB den FB ***SF_EQUIVALENT_REAL*** (→ Seite 285) einsetzen!

Für die Extended-Seite des ExtendedControllers endet der FB-Name mit "_E".

Symbol in CODESYS:



Beschreibung

2330

PERIOD misst die Frequenz und die Periodendauer (Zykluszeit) in [µs] am angegebenen Kanal (für alle Eingänge zugelassen). Maximale Eingangsfrequenz → Datenblatt.

Der FB misst die Frequenz und die Zykluszeit des am gewählten Kanal (CHANNEL) anstehenden Signals. Zur Berechnung werden alle positiven Flanken ausgewertet und der Mittelwert über die Anzahl der angegebenen Perioden (PERIODS) gebildet.

Bei niedrigen Frequenzen kommt es mit ***FREQUENCY*** (→ Seite 300) zu Ungenauigkeiten. Um dieses zu umgehen, kann PERIOD genutzt werden. Die Zykluszeit wird direkt in [µs] ausgegeben.

Der maximale Messbereich beträgt ca. 15 min.

> Am selben Eingang diesen FB **nicht** gemeinsam mit einem der folgenden FBs nutzen!
> - ***FAST_COUNT*** (→ Seite 298)
> - ***FREQUENCY*** (→ Seite 300)
> - ***FREQUENCY_PERIOD*** (→ Seite 302)
> - ***INC_ENCODER*** (→ Seite 304)
> - ***PERIOD_RATIO*** (→ Seite 308)
> - ***PHASE*** (→ Seite 310)

Parameter der Eingänge

2600

Parameter	Datentyp	Beschreibung
INIT	BOOL	FALSE ⇨ TRUE (Flanke): Baustein wird initialisiert FALSE: im weiteren Programmablauf
CHANNEL	BYTE	Nummer des schnellen Eingangskanals (0...15) 0...15 für die Eingänge I00...I15 Für den FB xxx**_E** (falls vorhanden) gilt: 0...15 für die Eingänge I00_E...I15_E
PERIODS	BYTE	Anzahl der Perioden, über die gemittelt wird (1...16) 0 : Ausgänge C und F werden nicht aktualisiert > 16 : wird auf 16 limitiert

Parameter der Ausgänge

375

Parameter	Datentyp	Beschreibung
C	DWORD	Zykluszeit der erfassten Perioden in [µs] zulässig = 200...10 000 000 = 0xC8...0x989680
F	REAL	Frequenz des Eingangssignals in [Hz]
ET	TIME	Verstrichene Zeit seit der letzten positiven Flanke am Eingang (nutzbar bei sehr langsamen Signalen)

PERIOD_RATIO

14018

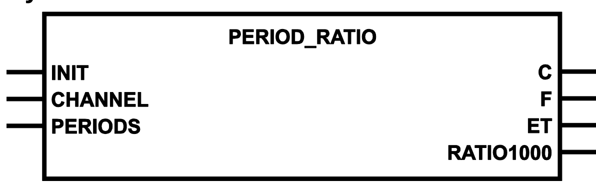
Baustein-Typ = Funktionsbaustein (FB)

Baustein ist enthalten in Bibliothek `ifm_CR7132_Vxxyyzz.LIB`

> Für Sicherheitssignale die Messwerte →diversitär auswerten!
> Zusätzlich zu diesem FB den FB ***SF_EQUIVALENT_REAL*** (→ Seite 285) einsetzen!

Für die Extended-Seite des ExtendedControllers endet der FB-Name mit "_E".

Symbol in CODESYS:



Beschreibung

2332

PERIOD_RATIO misst die Frequenz und die Periodendauer (Zykluszeit) in [µs] über die angegebenen Perioden am angegebenen Kanal (für alle Eingänge zugelassen). Zusätzlich wird das Puls-/Periode-Verhältnis in [‰] angegeben. Maximale Eingangsfrequenz → Datenblatt.

Dieser FB misst die Frequenz und die Zykluszeit des am gewählten Kanal (CHANNEL) anstehenden Signals. Zur Berechnung werden alle positiven Flanken ausgewertet und der Mittelwert über die Anzahl der angegebenen Perioden (PERIODS) gebildet. Zusätzlich wird das Puls-/Periode-Verhältnis in [‰] angegeben.
Zum Beispiel: Bei einem Signalverhältnis von 25 ms High-Pegel und 75 ms Low-Pegel, wird der Wert RATIO1000 von 250 ‰ ausgegeben.

Bei niedrigen Frequenzen kommt es mit ***FREQUENCY*** (→ Seite 300) zu Ungenauigkeiten. Um dieses zu umgehen, kann PERIOD_RATIO genutzt werden. Die Zykluszeit wird direkt in [µs] ausgegeben.

Der maximale Messbereich beträgt ca. 15 min.

> Am selben Eingang diesen FB **nicht** gemeinsam mit einem der folgenden FBs nutzen!
> - ***FAST_COUNT*** (→ Seite 298)
> - ***FREQUENCY*** (→ Seite 300)
> - ***FREQUENCY_PERIOD*** (→ Seite 302)
> - ***INC_ENCODER*** (→ Seite 304)
> - ***PERIOD*** (→ Seite 306)
> - ***PHASE*** (→ Seite 310)

Parameter der Eingänge

2601

Parameter	Datentyp	Beschreibung
INIT	BOOL	FALSE ⇨ TRUE (Flanke): Baustein wird initialisiert FALSE: im weiteren Programmablauf
CHANNEL	BYTE	Nummer des schnellen Eingangskanals (0...15) 0...15 für die Eingänge I00...I15 Für den FB xxx_**E** (falls vorhanden) gilt: 0...15 für die Eingänge I00_E...I15_E
PERIODS	BYTE	Anzahl der Perioden, über die gemittelt wird (1...16) 0 : Ausgänge C und F werden nicht aktualisiert > 16 : wird auf 16 limitiert

Parameter der Ausgänge

14318

Parameter	Datentyp	Beschreibung
C	DWORD	Zykluszeit der erfassten Perioden in [µs] zulässig = 200...10 000 000 = 0xC8...0x989680
F	REAL	Frequenz des Eingangssignals in [Hz]
ET	TIME	Verstrichene Zeit seit dem letzten Zustandswechsel am Eingang (nutzbar bei sehr langsamen Signalen)
RATIO1000	WORD	Puls-/Periode-Verhältnis in [‰] Voraussetzungen: • Periodendauermessung • Impulsdauer ≥ 100 µs • Frequenz < 5 kHz Wert nicht für sicherheitsrelevante Funktionen verwenden!

PHASE

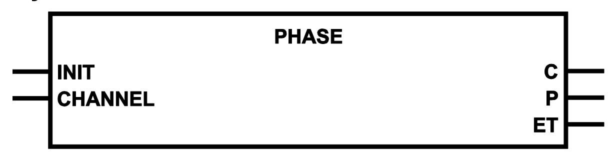
358

Baustein-Typ = Funktionsbaustein (FB)

Baustein ist enthalten in Bibliothek `ifm_CR7132_Vxxyyzz.LIB`

Für die Extended-Seite des ExtendedControllers endet der FB-Name mit "_E".

Symbol in CODESYS:



Beschreibung

2338

PHASE liest ein Kanalpaar mit schnellen Eingängen ein und vergleicht die Phasenlage der Signale.
Maximale Eingangsfrequenz → Datenblatt.

Dieser FB fasst jeweils ein Kanalpaar mit schnellen Eingängen zusammen, so dass die Phasenlage zweier Signale zueinander ausgewertet werden kann. Es kann eine Periodendauer bis in den Sekundenbereich ausgewertet werden.

Am selben Eingang diesen FB **nicht** gemeinsam mit einem der folgenden FBs nutzen!
- *FAST_COUNT* (→ Seite 298)
- *FREQUENCY* (→ Seite 300)
- *FREQUENCY_PERIOD* (→ Seite 302)
- *INC_ENCODER* (→ Seite 304)
- *PERIOD* (→ Seite 306)
- *PERIOD_RATIO* (→ Seite 308)

Parameter der Eingänge

2339

Parameter	Datentyp	Beschreibung
INIT	BOOL	TRUE (nur 1 Zyklus lang): Baustein und Schnittstelle werden initialisiert FALSE: im weiteren Programmablauf
CHANNEL	BYTE	Nummer des Eingangskanal-Paares (0...7) 0 = Kanalpaar 0 = Eingänge I00 + I01 ... 7 = Kanalpaar 7 = Eingänge I14 + I15 Für den FB xxx**_E** (falls vorhanden) gilt: 0 = Kanalpaar 0 = Eingänge I00_E + I01_E ... 7 = Kanalpaar 7 = Eingänge I14_E + I15_E

Parameter der Ausgänge

363

Parameter	Datentyp	Beschreibung
C	DWORD	Periodendauer des Signals am ersten Eingang des Kanalpaares in [µs]
P	INT	Winkel der Phasenverschiebung gültige Messung = 1...358 °
ET	TIME	Verstrichene Zeit seit der letzten positiven Flanke am zweiten Impulseingang des Kanalpaares

6.2.13 Bausteine: Ausgangsfunktionen allgemein

Inhalt

10462

Für dieses Gerät können Sie die Funktionsweise von einigen oder von allen Ausgängen einstellen.
Hier zeigen wir Ihnen geeignete Bausteine dazu.

SET_OUTPUT_MODE

13029

Baustein-Typ = Funktionsbaustein (FB)

Baustein ist enthalten in Bibliothek `ifm_CR7132_Vxxyyzz.LIB`

Für die Extended-Seite des ExtendedControllers endet der FB-Name mit "_E".

Symbol in CODESYS:

```
SET_OUTPUT_MODE
ENABLE            ERROR
MODE
CHANNEL
DIAGNOSTICS
PROTECTION
CURRENT_RANGE
SAFETY
```

Beschreibung

13031
12094

SET_OUTPUT_MODE setzt die Betriebsart des gewählten Ausgangskanals.
Zugelassene Betriebsarten (→ Datenblatt):

MODE	Konfig.-Wert	
	hex	dez
---	---	---
OUT_DIGITAL_H (plus)	0001	1
OUT_DIGITAL_L (minus)	0002	2

CURRENT_RANGE	Konfig.-Wert	
	hex	dez
keine Strommessung für MODE = 0	00	0
Strommessung 2 A (3 A) für MODE = 1 oder 2	01	1
Strommessung 4 A für MODE = 1 oder 2	02	2

Im laufenden Betrieb sollte die Betriebsart nicht geändert werden.

15672

HINWEIS

Soll im laufenden Betrieb am FB OUTPUT_BRIDGE der Messbereich für ACTUAL_CURRENT (auf 4 A) umgeschaltet werden?

- ► Den FB SET_OUTPUT_MODE in der Init-Phase **vor** dem FB OUTPUT_BRIDGE aufrufen!
- ► Beim Aufruf des FB SET_OUTPUT_MODE am FB OUTPUT_BRIDGE den Parameter DIRECTION berücksichtigen!
  Das Umschalten des Messbereichs ist nur für den in B(H) betriebenen Ausgang zulässig, nämlich:

DIRECTION	H-Bridge	Ausgang
0	1 2	Q01(_E) Q09(_E)
1	1 2	Q03(_E) Q11(_E)

HINWEIS

Bei Strommessung: die Filtereinstellungen Qxx_FILTER wirken sich auf die Diagnosezeit aus.

► Qxx_FILTER für die sicheren Ausgänge auf den voreingestellten Werten belassen!

Falls Versorgungsspannung < 7,3 V: an den Ausgängen keine Kurzschlusserkennung möglich!

Falls der Ausgangskanal als Sicherheitsausgang konfiguriert ist (SAFETY=TRUE):

- CURRENT_RANGE auf > 0 einstellen!
  Andernfalls ist keine Überwachung möglich.
- Wenn derselbe Ausgangskanal während der Laufzeit von einem der folgenden FBs aufgerufen wird, dann wertet die Steuerung dies als schweren Fehler!
  - OUTPUT_CURRENT_CONTROL
  - OUTPUT_BRIDGE
  - PWM1000
  - SAFETY_SWITCH



Parameter der Eingänge

13032

Parameter	Datentyp	Beschreibung
ENABLE	BOOL	FALSE ⇨ TRUE (Flanke): Baustein initialisieren (nur 1 Zyklus) > Baustein-Eingänge lesen TRUE: Baustein ausführen FALSE: Baustein wird nicht ausgeführt > Baustein-Eingänge sind nicht aktiv > Baustein-Ausgänge sind nicht spezifiziert
MODE	WORD	Betriebsart des Ausgangskanals CHANNEL: 1 = 0x0001 OUT_DIGITAL_H 2 = 0x0002 OUT_DIGITAL_L
CHANNEL	BYTE	Nummer des Ausgangskanals (0...15) 0...15 für die Ausgänge Q00...Q15 Für den FB xxx**_E** (falls vorhanden) gilt: 0...31 für die Ausgänge Q00_E...Q31_E
DIAGNOSTICS	BOOL	TRUE: Kanal mit Diagnosefunktion nur wirksam für OUT_DIGITAL_H • Leiterbruch bei Ausgängen mit Strommessung: wenn Strom < 25 mA für ≥ 66 ms • Leiterbruch bei Ausgängen ohne Strommessung: wenn Ausgangsspannung > 22 % VBBx für ≥ 66 ms • Überlast bei Strom > 112,5 % des Messbereichs für ≥ 66 ms • Kurzschluss bei Ausgangsspannung < 88 % VBBx für ≥ 66 ms FALSE: Kanal ohne Diagnosefunktion
PROTECTION	BOOL	TRUE: Schutz vor Überlast nur für OUT_DIGITAL_H UND Ausgang mit Strommessung Bei Erkennen von Überlast oder Kurzschluss schaltet der Ausgang für 1 s aus und dann wieder ein. FALSE: Funktion wird nicht ausgeführt
CURRENT_RANGE	BYTE	Strommessbereich im Ausgangskanal CHANNEL: 0 = 0x00 CURRENT_RANGE_NONE (Aus) nur für Ausgang ohne Strommessung 1 = 0x01 CURRENT_RANGE1 2 A für OUT_DIGITAL_H 2 = 0x02 CURRENT_RANGE2 4 A für OUT_DIGITAL_H
SAFETY	BOOL	CHANNEL soll als Sicherheitskanal betrieben werden Nur zulässig für Betriebsart OUT_DIGITAL_H Voraussetzung für SAFETY=TRUE: • DIAGNOSTICS=TRUE • PROTECTION=TRUE Andernfalls ⇨ Parametrierfehler! Sobald SAFETY=TRUE, ist Folgendes für diesen Kanal nicht mehr zulässig: • SAFETY=FALSE • ändern von MODE Andernfalls ⇨ schwerer Fehler! TRUE: CHANNEL ist Sicherheitskanal FALSE: CHANNEL ist Standardkanal

Parameter der Ausgänge

13035

Parameter	Datentyp	Beschreibung
ERROR	DWORD	Fehler-Code aus diesem FB-Aufruf → ***Fehler-Codes*** (→ Seite 392) (mögliche Meldungen → folgende Tabelle)

Mögliche Ergebnisse für ERROR (n=beliebiger Wert):

Der 32-Bit-Fehler-Code besteht aus vier 8-Bit-Werten (DWORD).

4. Byte	3. Byte	2. Byte	1. Byte
Fehlerklasse	anwendungsspezifischer Fehler-Code	Fehlerquelle	Fehlerursache

Wert [hex]	Beschreibung
00 00 00 00	kein Fehler
02 00 00 F8	falscher Parameter ⇨ schwerer Fehler

6.2.14 Bausteine: Ausgangswerte sicher verarbeiten

Inhalt

12718

Hier finden Sie Funktionsbausteine zum sichern Verarbeiten von sicherheitsrelevanten Ausgangswerten.

Die hier aufgeführten Sicherheits-FBs sind für SIL CL 2 nach IEC 62061-1 zertifiziert.

8340

⚠ WARNUNG

Für die sichere Funktion der Anwendungsprogramme, die vom Anwender erstellt werden, ist dieser selbst verantwortlich. Bei Bedarf muss er zusätzlich entsprechend der nationalen Vorschriften eine Abnahme durch entsprechende Prüf- und Überwachungsorganisationen durchführen lassen.

Legende zu den Ein- und Ausgängen der Sicherheits-FBs 'SF_...'

12700

- Alle Ein- und Ausgänge der Sicherheits-FBs 'SF_...' werden beim Hochlaufen (Booten) der Steuerung mit Initialwerten belegt. So werden undefinierte Zustände der FBs verhindert. Die Initialwerte führen immer zum sicheren Zustand der Steuerung.
- Alle Sicherheits-FBs arbeiten nur, wenn der Eingang ENABLE=TRUE ist.
- Der Ausgang READY=TRUE gibt an, dass der FB aktiv ist.
- Der Ausgang SAFETYDEMAND=TRUE signalisiert, dass Handlungsbedarf durch den Bediener erforderlich ist.
- Der Eingang S_SAFETYACTIVE=TRUE signalisiert dem FB, dass der relevante Prozess im Sicherheitsmodus ist.
- Der Ausgang RESETREQUEST=TRUE fordert vom Bediener:
  • gegebenenfalls die falschen Eingangssignale beheben
  • mit RESET-Signal bestätigen, dass der FB den sicheren Prozess fortsetzen darf
  Andernfalls verbleibt der FB im Warten- oder Fehler-Zustand.
  ℹ FBs ohne Eingang RESET: der Ausgang RESETREQUEST ist ohne Funktion.
- Der Eingang RESET=TRUE (nur 1 Zyklus) signalisiert dem FB:
  • den sicheren Prozess fortsetzen, sofern die Fehler beseitigt sind.
  Dauerhafter RESET führt zum Fehler.
- Der Ausgang ERROR=TRUE signalisiert einen Fehler.
  Der Fehler-Code erscheint im Ausgang DIAGCODE mit Werten ab 0xC000.
- Der Ausgang DIAGCODE liefert permanent Informationen über den Status des FBs.

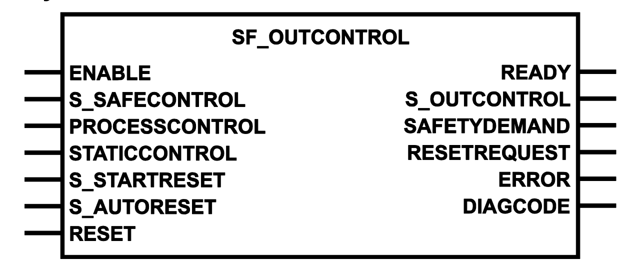
SF_OUTCONTROL

12720

Baustein-Typ = Funktionsbaustein (FB)

Baustein ist enthalten in Bibliothek `ifm_SafetyPLCopen_Vxxyyzz.lib`

Symbol in CODESYS:



Beschreibung

12722

SF_OUTCONTROL kontrolliert einen sicheren Ausgang mit einem Signal aus der funktionellen Anwendung und einem sicheren Signal mit optionaler Anlaufsperre.

Parameter der Eingänge

12723

Parameter	Datentyp	Beschreibung
ENABLE	BOOL	TRUE: Baustein ausführen FALSE: Baustein wird nicht ausgeführt > Baustein-Eingänge sind nicht aktiv > Baustein-Ausgänge sind nicht spezifiziert
S_SAFECONTROL	BOOL	Steuersignal des vorausgehenden sicheren FB TRUE: sicheres Steuersignal ist eingeschaltet FALSE (= Initialwert): sicheres Steuersignal ist ausgeschaltet
PROCESSCONTROL	BOOL	Steuersignal der funktionellen Anwendung TRUE: S_OUTCONTROL auf TRUE setzen FALSE (=Initialwert): S_OUTCONTROL auf FALSE setzen
STATICCONTROL	BOOL	Zusätzliche Bedingung für PROCESSCONTROL TRUE: Keine steigende Flanke am Eingang PROCESSCONTROL erforderlich nach Aktivierung des FB oder nach steigender Flanke S_SAFECONTROL FALSE (= Initialwert): steigende Flanke am Eingang PROCESSCONTROL erforderlich nach Aktivierung des FB oder nach steigender Flanke S_SAFECONTROL
S_STARTRESET	BOOL	Nach dem Aktivieren des FB ... TRUE: ... erfolgt automatisch ein Reset. FALSE (= Initialwert): ... ist ein manueller Reset erforderlich.
S_AUTORESET	BOOL	Nach dem Zurücksetzen des Sicherheitsschalters ... TRUE: ... erfolgt automatisch ein Reset. FALSE (= Initialwert): ... ist ein manueller Reset erforderlich.
RESET	BOOL	TRUE (nur 1 Zyklus lang): • Bestätigung: sicherer Zustand ist erfüllt • Bestätigung: Fehler ist behoben sonst: diese Funktion wird nicht ausgeführt (=Initialwert)

Parameter der Ausgänge

12724

Parameter	Datentyp	Beschreibung
READY	BOOL	TRUE: Funktionsbaustein ist aktiv Die Werte an den FB-Ausgängen sind gültig FALSE (= Initialwert): Funktionsbaustein ist nicht aktiv
S_OUTCONTROL	BOOL	TRUE: Angeschlossener Aktuator ist aktiv FALSE (= Initialwert): Angeschlossener Aktuator ist nicht aktiv
SAFETYDEMAND	BOOL	TRUE: Sicherheitsfunktion ist angefordert Aktion des Maschinenführers erforderlich FALSE (= Initialwert): Sicherheitsfunktion nicht angefordert
RESETREQUEST	BOOL	Ein Reset ist erforderlich, damit der FB weiterarbeiten kann TRUE: Reset ist erforderlich FALSE (= Initialwert): kein Reset erforderlich
ERROR	BOOL	TRUE: ein Fehler ist aufgetreten FB ist im Fehlerzustand FALSE (= Initialwert): kein Fehler aufgetreten
DIAGCODE	WORD	Informationen zum aktuellen Zustand des FBs (mögliche Meldungen → folgende Tabelle)

Mögliche Ergebnisse für DIAGCODE:

Wert [hex]	Beschreibung
0000	(Initialwert) Funktionsbaustein ist nicht aktiv
8000	Sicherheitsausgang ist freigegeben
8001	Funktionsbaustein ist bereit: READY=TRUE S_STARTRESET=FALSE FB wartet auf RESET=TRUE
8002	Funktionsbaustein ist bereit: READY=TRUE FB wartet auf Sicherheitseingang=TRUE
8003	nach S_SAFECONTROL=FALSE FB wartet auf RESET=TRUE
8010	PROCESSCONTROL ist nicht aktiv
C001	im Zustand 8001 ist RESET-Signal statisch FB wartet auf RESET=FALSE
C002	im Zustand 8003 ist RESET-Signal statisch FB wartet auf RESET=FALSE
C010	im Zustand 8010 ist PROCESSCONTROL-Signal statisch FB wartet auf PROCESSCONTROL=FALSE
C111	Fehler im Ablauf im Zustand 8001: gleichzeitig RESET + PROCESSCONTROL ⇨ TRUE
C211	Fehler im Ablauf im Zustand 8003: gleichzeitig RESET + PROCESSCONTROL ⇨ TRUE

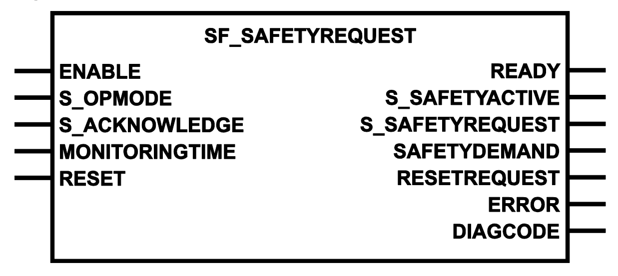
SF_SAFETYREQUEST

12692

Baustein-Typ = Funktionsbaustein (FB)

Baustein ist enthalten in Bibliothek `ifm_SafetyPLCopen_Vxxyyzz.lib`

Symbol in CODESYS:



Beschreibung

12695

SF_SAFETYREQUEST stellt eine Schnittstelle zu einem allgemeinen Aktuator mit Rückkanal zur Verfügung zu folgenden Zwecken:

- den Aktuator in den sicheren Zustand setzen
- überwachen, ob der Aktuator den sicheren Zustand in der vorgegebenen Zeit eingenommen hat.

Parameter der Eingänge

12696

Parameter	Datentyp	Beschreibung
ENABLE	BOOL	TRUE: Baustein ausführen FALSE: Baustein wird nicht ausgeführt > Baustein-Eingänge sind nicht aktiv > Baustein-Ausgänge sind nicht spezifiziert
S_OPMODE	BOOL	für den sicheren Aktuator angeforderter Betriebsmodus TRUE: non-safe Betriebsmodus angefordert FALSE (= Initialwert): sicherer Modus angefordert
S_ACKNOWLEDGE	BOOL	Rückmeldung des Betriebsmodus vom sicheren Aktuator TRUE: sicherer Modus FALSE (= Initialwert): non-safe Betriebsmodus
MONITORTIME	TIME	maximal zulässige Antwortzeit zwischen der Sicherheitsanfrage (S_OPMODE=FALSE) und der Bestätigung des Aktuators (S_ACKNOWLEDGE=TRUE) Initialwert = T#0ms
RESET	BOOL	TRUE (nur 1 Zyklus lang): • Bestätigung: sicherer Zustand ist erfüllt • Bestätigung: Fehler ist behoben sonst: diese Funktion wird nicht ausgeführt (=Initialwert)

Parameter der Ausgänge

12697

Parameter	Datentyp	Beschreibung
READY	BOOL	TRUE: Funktionsbaustein ist aktiv Die Werte an den FB-Ausgängen sind gültig FALSE (= Initialwert): Funktionsbaustein ist nicht aktiv
S_SAFETYACTIVE	BOOL	Die Maschine ist im Sicherheitsmodus: z.B. Bewegungsgeschwindigkeit ist sicher reduziert. Nur bei aktivem Sicherheitsmodus darf die Freigabetaste den Sicherheitsausgang aktivieren. TRUE: der Sicherheitsmodus ist aktiv Die Freigabetaste kann den Sicherheitsausgang aktivieren. FALSE (= Initialwert): der Sicherheitsmodus ist nicht aktiv
S_SAFETYREQUEST	BOOL	Anforderung, den Aktuator in den sicheren Zustand zu versetzen TRUE: keinen sicheren Zustand angefordert FALSE (= Initialwert): sicheren Zustand angefordert
SAFETYDEMAND	BOOL	TRUE: Sicherheitsfunktion ist angefordert Aktion des Maschinenführers erforderlich FALSE (= Initialwert): Sicherheitsfunktion nicht angefordert
RESETREQUEST	BOOL	Ein Reset ist erforderlich, damit der FB weiterarbeiten kann TRUE: Reset ist erforderlich FALSE (= Initialwert): kein Reset erforderlich
ERROR	BOOL	TRUE: ein Fehler ist aufgetreten FB ist im Fehlerzustand FALSE (= Initialwert): kein Fehler aufgetreten
DIAGCODE	WORD	Informationen zum aktuellen Zustand des FBs (mögliche Meldungen → folgende Tabelle)

Mögliche Ergebnisse für DIAGCODE:

Wert [hex]	Beschreibung
0000	(Initialwert) Funktionsbaustein ist nicht aktiv
8000	der Aktuator ist im sicheren Modus
8001	Sicherheitseingang bereits bei FB-Start aktiv
8002	sicherer Modus angefordert
8003	FB wartet auf Aktuator im sicheren Modus MONITORTIME überwacht den Vorgang
8005	Ein Fehlerzustand wurde zurückgesetzt. ▶ S_OPMODE=TRUE setzen für FB-Initialisierung!
8012	Aktuator ist im sicheren Modus
C002	Fehler im Ablauf, z.B. Reihenfolge FB wartet auf RESET
C003	Überwachungszeit abgelaufen (FB wartet auf Aktuator im non-safe-Modus)
C004	im Zustand C002 ist RESET-Signal statisch FB wartet auf RESET=FALSE

Wert [hex]	Beschreibung
C005	im Zustand C003 ist RESET-Signal statisch FB wartet auf RESET=FALSE

6.2.15 Bausteine: PWM-Funktionen

Inhalt

13758

Hier finden Sie **ifm**-Bausteine, um die Ausgänge mit Pulsweitenmodulation (PWM) betreiben zu können.

OUTPUT_BRIDGE

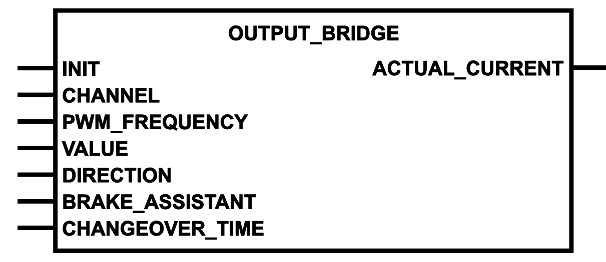
2198

Baustein-Typ = Funktionsbaustein (FB)

Baustein ist enthalten in Bibliothek `ifm_CR7132_Vxxyyzz.LIB`

Für die Extended-Seite des ExtendedControllers endet der FB-Name mit "_E".

Symbol in CODESYS:



Beschreibung

14023
19298

OUTPUT_BRIDGE organisiert das Ansteuern der H-Brücken an den PWM-Kanälen.

Der FB dient zur einfachen Verwendung der Ausgänge als H-Brücke. Dazu werden jeweils zwei aufeinander folgende Ausgangskanäle mit minus-schaltendem Treiber zu einer Brücke zusammengefasst. Ist DIRECTION = FALSE, wird beim ersten Ausgang der plus-schaltende Treiber über ein PWM-Signal angesteuert und der minus-schaltende Treiber des zweiten Ausgangs ist durchgeschaltet.

HINWEIS

Bei Einsatz der H-Brücke wird die Stromregelung nicht unterstützt.

Ausgänge, die im PWM-Modus betrieben werden, unterstützen keine Diagnosefunktionen und es werden keine ERROR-Merker gesetzt.

Die Überlastschutzfunktion ist in diesem Modus nicht aktiv!
Die Überlastschutzfunktion wird durch OUTPUT_BRIDGE zurückgesetzt.

Bei VALUE = 0 wird der Ausgang nicht komplett deaktiviert. Prinzipbedingt wird der Ausgang für die Dauer eines Timer-Ticks des PWM-Timers aktiv sein (typisch ca. 50 µs).

► FB in jedem SPS-Zyklus aufrufen!

Lage der als H-Brücke verwendbaren Ausgangskanäle → Datenblatt.

15672

HINWEIS

Soll im laufenden Betrieb am FB OUTPUT_BRIDGE der Messbereich für ACTUAL_CURRENT (auf 4 A) umgeschaltet werden?

- ► Den FB SET_OUTPUT_MODE in der Init-Phase **vor** dem FB OUTPUT_BRIDGE aufrufen!
- ► Beim Aufruf des FB SET_OUTPUT_MODE am FB OUTPUT_BRIDGE den Parameter DIRECTION berücksichtigen!
  Das Umschalten des Messbereichs ist nur für den in B(H) betriebenen Ausgang zulässig, nämlich:

DIRECTION	H-Bridge	Ausgang
0	1 2	Q01(_E) Q09(_E)
1	1 2	Q03(_E) Q11(_E)

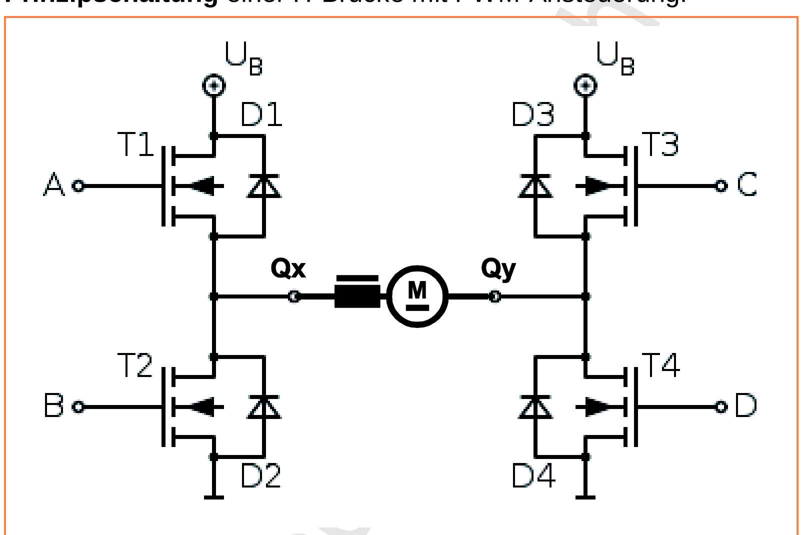
Ein Ausgang mit H-Brücke darf **nicht** als sicherer Ausgang konfiguriert werden!
Andernfalls meldet das Laufzeitsystem einen schweren Fehler.

Prinzip der H-Brücke

9990
16411

Hier sehen Sie, wie eine H-Brücke am **ifm**-Controller via PWM-Ausgängen betrieben werden kann.

Prinzipschaltung einer H-Brücke mit PWM-Ansteuerung:



T1 und T2 bilden zusammen z.B. den Ausgang Qx.
Genauso bilden T3 und T4 z.B. den Ausgang Qy.
Dadurch werden nur zwei Anschlüsse für den DC-Motor benötigt.

Programm-Beispiel:

```
VAR
    Init1: BOOL:=TRUE;
    CycleTime:DWORD;
    MaxCycleTime:DWORD;
    ResetMax:BOOL;

    DownloadID: CAN1_DOWNLOADID;

(******************************************************************************
CHANNEL = 1:    Motor between OUT01 (Pin17) and OUT03(Pin15)
CHANNEL = 2:    Motor between OUT09 (Pin03) and OUT11(Pin05)
******************************************************************************)
    H_BRIDGE: OUTPUT_BRIDGE;
    PWM_value: WORD := 100;         (* current PWM value - VALUE = 0...1000 *)
    H_direction: BOOL;              (* TRUE = counter clockwise; FALSE = clockwise *)
    H_current: WORD;                (* output current in mA *)
    changeover_time: WORD := 500;   (* Space time [ms] during which the motor is not triggered
                                       (> 10 ms) in the case of a change of the rotational direction. *)
END_VAR
```

```
                          H_BRIDGE
                 +----------------------------------+
                 |        OUTPUT_BRIDGE             |
          Init1--|INIT                ACTUAL_CURRENT|---H_current
              1--|CHANNEL                           |
            250--|PWM_FREQUENCY                     |
      PWM_value--|VALUE                             |
    H_direction--|DIRECTION                         |
          FALSE--|BRAKE_ASSISTANT                   |
           2000--|CHANGEOVER_TIME                   |
                 +----------------------------------+
```

Parameter der Eingänge

2204

Parameter	Datentyp	Beschreibung
INIT	BOOL	TRUE (nur 1 Zyklus lang): Baustein wird initialisiert FALSE: im weiteren Programmablauf
CHANNEL	BYTE	Name des Ausgangspaares: 1 = Brücke 1 an Q01 + Q03 2 = Brücke 2 an Q09 + Q11 Für den FB xxx_**E** (falls vorhanden) gilt: 1 = Brücke 1 an Q01_E + Q03_E 2 = Brücke 2 an Q09_E + Q11_E
PWM_FREQUENCY	WORD	PWM-Frequenz [Hz] für die Last am Ausgang > FB begrenzt den Wert auf 20...2 000 = 0x0014...0x07D0 ▶ Für Änderung des Werts: FB neu initialisieren!
VALUE	WORD	PWM-Wert (Puls-Periode-Verhältnis) in [‰] zulässig = 0...1 000 = 0x0000...0x03E8 Werte > 1 000 gelten als = 1 000
DIRECTION	BOOL	Drehrichtung des Motors: TRUE: entgegen Uhrzeigersinn (ccw): Brücke 1: Stromfluss Q01(_E) ⇦ Q03(_E) Brücke 2: Stromfluss Q09(_E) ⇦ Q11(_E) FALSE: im Uhrzeigersinn (cw): Brücke 1: Stromfluss Q01(_E) ⇨ Q03(_E) Brücke 2: Stromfluss Q09(_E) ⇨ Q11(_E)
BRAKE_ASSISTANT	BOOL	TRUE: Beim Wechsel der Drehrichtung: FB schaltet beide Ausgänge gegen Masse, zwecks Bremswirkung am Motor, solange CHANGEOVER_TIME läuft. FALSE: Funktion wird nicht ausgeführt
CHANGEOVER_TIME	WORD	Pausezeit in [ms], während der bei einem Wechsel der Drehrichtung der Motor nicht angesteuert wird (≥ Zykluszeit, mindestens 10 ms)

Parameter der Ausgänge

2205

Parameter	Datentyp	Beschreibung
ACTUAL_CURRENT	WORD	Ausgangsstrom in [mA]

OUTPUT_CURRENT

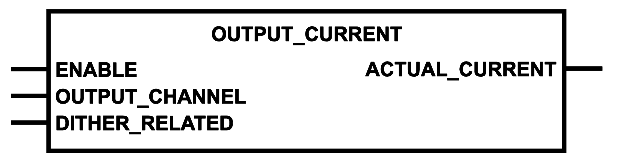
382

Baustein-Typ = Funktionsbaustein (FB)

Baustein ist enthalten in Bibliothek `ifm_CR7132_Vxxyyzz.LIB`

Für die Extended-Seite des ExtendedControllers endet der FB-Name mit "_E".

Symbol in CODESYS:



Beschreibung

385

OUTPUT_CURRENT dient dem Messen des Stroms (optional: Mittelung über Dither-Periode) an einem Ausgangskanal.

Der FB liefert den aktuellen Ausgangsstrom, wenn die Ausgänge als PWM-Ausgänge oder als plus-schaltend benutzt werden. Die Strommessung erfolgt innerhalb des Gerätes, es werden also keine externen Messwiderstände benötigt.

Parameter der Eingänge

17894

Parameter	Datentyp	Beschreibung
ENABLE	BOOL	TRUE: Baustein ausführen FALSE: Baustein wird nicht ausgeführt > Baustein-Eingänge sind nicht aktiv > Baustein-Ausgänge sind nicht spezifiziert
OUTPUT_CHANNEL	BYTE	Nummer des stromgeregelten Ausgangskanals (0...15) 0...15 für die Ausgänge Q00...Q15 Für den FB xxx**_E** (falls vorhanden) gilt: 0...15 für die Ausgänge Q00_E...Q15_E
DITHER_RELATED	BOOL	Strom wird ermittelt als Mittelwert über... TRUE: eine Dither-Periode FALSE: eine PWM-Periode

Parameter der Ausgänge

387

Parameter	Datentyp	Beschreibung
ACTUAL_CURRENT	WORD	Ausgangsstrom in [mA]

OUTPUT_CURRENT_CONTROL

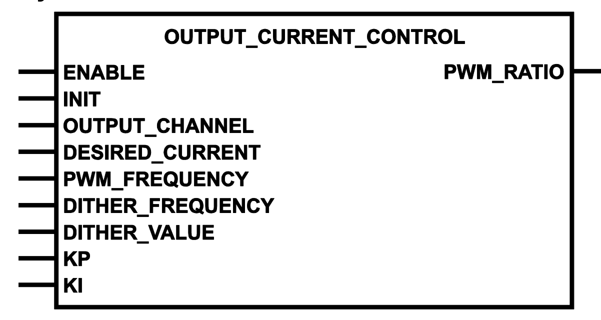
2196

Baustein-Typ = Funktionsbaustein (FB)

Baustein ist enthalten in Bibliothek `ifm_CR7132_Vxxyyzz.LIB`

Für die Extended-Seite des ExtendedControllers endet der FB-Name mit "_E".

Symbol in CODESYS:



Beschreibung

2200

OUTPUT_CURRENT_CONTROL arbeitet als Stromregler für die PWM-Ausgänge.

Der Regler regelt in Abhängigkeit der Periodendauer des PWM Signals. Die beiden Anstellparameter KI und KP repräsentieren den I- und den P-Anteil des Reglers. Zur Ermittlung der besten Einstellung des Reglers bietet sich als Startwert an, KI = 50 und KP = 50 zu setzen. Je nach gewünschtem Reglerverhalten können die Werte schrittweise vergrößert (Regler wird härter / schneller) oder verkleinert (Regler wird schwächer / langsamer) werden.

Bei Sollwert DESIRED_CURRENT=0 wird der Ausgang innerhalb von etwa 100 ms auf 0 mA heruntergeregelt , wobei die Anstellparameter ignoriert werden.

Je nach eingesetzter Steuerungs-Hardware ist ein unterschiedliches Teach-Verhalten zu beachten.

HINWEIS

▶ Bei der Definition des Parameters DITHER_VALUE darauf achten, dass das resultierende PWM-Ratio im Arbeitsbereich der Regelung zwischen 0...1000 ‰ bleibt:
- PWM-Ratio + DITHER_VALUE < 1000 ‰ und
- PWM-Ratio - DITHER_VALUE > 0 ‰.

Außerhalb dieses zulässigen Bereichs kann der im Parameter DESIRED_CURRENT angegebene Strom nicht erreicht werden.

> Bei aktiviertem Dither werden Änderungen an PWM_FREQUENCY, DITHER_VALUE und DITHER_FREQUENCY erst nach Ende der aktuellen Dither-Periode angewendet.

> Kann der im Parameter DESIRED_CURRENT angegebene Strom nicht erreicht werden, weil das PWM-Ratioverhältnis schon bei 100 % ist, wird das durch die Systemvariable ERROR_CONTROL_Qx angezeigt (x = Kanalnummer).

> Bei KI = 0 findet keine Regelung statt.

> Ergibt sich bei der Regelung ein PWM_RATIO = 0, wird der Ausgang nicht komplett deaktiviert. Prinzipbedingt wird der Ausgang für die Dauer eines Timer-Ticks des PWM-Timers aktiv sein (typisch ca. 50 µs).

Parameter der Eingänge

2201

Parameter	Datentyp	Beschreibung
ENABLE	BOOL	TRUE: Baustein ausführen FALSE: Baustein wird nicht ausgeführt > Baustein-Eingänge sind nicht aktiv > Baustein-Ausgänge sind nicht spezifiziert
INIT	BOOL	TRUE (nur 1 Zyklus lang): Baustein wird initialisiert FALSE: im weiteren Programmablauf
OUTPUT_CHANNEL	BYTE	Nummer des stromgeregelten Ausgangskanals (0...15) 0...15 für die Ausgänge Q00...Q15 Für den FB xxx_**E** (falls vorhanden) gilt: 0...15 für die Ausgänge Q00_E...Q15_E
DESIRED_CURRENT	WORD	Stromsollwert des Ausgangs in [mA] zulässig = 0...2 000 / 0...4 000 (abhängig vom Ausgang und der Konfiguration)
PWM_FREQUENCY	WORD	PWM-Frequenz [Hz] für die Last am Ausgang > FB begrenzt den Wert auf 20...2 000 = 0x0014...0x07D0 ▶ Für Änderung des Werts: FB neu initialisieren!
DITHER_FREQUENCY	WORD	Dither-Frequenz in [Hz] Wertebereich = 0...FREQUENCY / 2 FREQUENCY / DITHER_FREQUENCY muss geradzahlig sein! Alle anderen Werte erhöht der FB auf den nächst passenden Wert.
DITHER_VALUE	WORD	Spitze-Spitze-Wert des Dithers in [‰] zulässig = 0...1 000 = 0x0000...0x03E8
KP	BYTE	Proportional-Anteil des Ausgangssignals
KI	BYTE	Integral-Anteil des Ausgangssignals bei KI = 0 keine Regelung

Parameter der Ausgänge

2202

Parameter	Datentyp	Beschreibung
PWM_RATIO	WORD	Zu Kontrollzwecken: Anzeige PWM-Tastverhältnis 0...999 ‰

PWM1000

326

Baustein-Typ = Funktionsbaustein (FB)

Baustein ist enthalten in Bibliothek `ifm_CR7132_Vxxyyzz.LIB`

Für die Extended-Seite des ExtendedControllers endet der FB-Name mit "_E".

Symbol in CODESYS:

PWM1000
INIT
FREQUENCY
CHANNEL
VALUE
CHANGE
DITHER_VALUE
DITHER_FREQUENCY

Beschreibung

14020
2311

PWM1000 initialisiert und parametriert einen PWM-fähigen Ausgang.

Der FB ermöglicht eine einfache Anwendung der PWM-Funktion im Gerät. Für jeden Kanal kann jeweils eine eigene PWM-Frequenz, das Puls-Periode-Verhältnis und der Dither eingestellt werden.

Die PWM-Frequenz FREQUENCY kann direkt in [Hz] und das Puls-Periode-Verhältnis VALUE in 1 ‰-Schritten angegeben werden.

Bei VALUE = 0 wird der Ausgang nicht komplett deaktiviert. Prinzipbedingt wird der Ausgang für die Dauer eines Timer-Ticks des PWM-Timers aktiv sein (typisch ca. 50 µs).

Der FB wird einmalig für jeden Kanal in der Initialisierung des Anwendungsprogramms aufgerufen.

> **HINWEIS**
>
> Die PWM-Funktion bleibt solange gesetzt, bis an der Steuerung ein Hardware-Reset durchgeführt wurde ➩ Versorgungsspannung ausschalten und wieder einschalten.
>
> Bei hohen PWM-Frequenzen kann es systembedingt zu Differenzen kommen zwischen eingestelltem und ausgegebenem Ratio-Verhältnis.

Änderungen während der Laufzeit:

Immer, wenn Eingang INIT auf TRUE gesetzt ist, übernimmt der FB die angegebene PWM-Frequenz FREQUENCY.

Immer, wenn Eingang CHANGE auf TRUE gesetzt ist, übernimmt der FB den Wert ...
- VALUE nach der aktuellen PWM-Periode
- DITHER_VALUE nach der aktuellen Dither-Periode
- DITHER_FREQUENCY nach der aktuellen Dither-Periode

> Ein Ausgang mit PWM darf **nicht** als sicherer Ausgang konfiguriert werden!
> Andernfalls meldet das Laufzeitsystem einen schweren Fehler.

Parameter der Eingänge

2312

Parameter	Datentyp	Beschreibung
INIT	BOOL	TRUE (nur 1 Zyklus lang): Baustein wird initialisiert Übernahme neuer Wert von FREQUENCY FALSE: im weiteren Programmablauf
FREQUENCY	WORD	PWM-Frequenz in [Hz] > FB begrenzt den Wert auf 20...2 000 = 0x0014...0x07D0 ▶ Für Änderung des Werts: FB neu initialisieren!
CHANNEL	BYTE	Nummer des PWM-Ausgangskanals (0...15) 0...15 für die Ausgänge Q00...Q15 Für den FB xxx_**E** (falls vorhanden) gilt: 0...15 für die Ausgänge Q00_E...Q15_E
VALUE	WORD	PWM-Wert (Puls-Periode-Verhältnis) in [‰] zulässig = 0...1 000 = 0x0000...0x03E8 Werte > 1 000 gelten als = 1 000
CHANGE	BOOL	TRUE: Übernahme neuer Wert von ... • VALUE: nach der aktuellen PWM-Periode • DITHER_VALUE: nach der aktuellen Dither-Periode • DITHER_FREQUENCY: nach der aktuellen Dither-Periode FALSE: geänderter PWM-Wert hat keinen Einfluss auf den Ausgang
DITHER_VALUE	WORD	Spitze-Spitze-Wert des Dithers in [‰] zulässig = 0...1 000 = 0x0000...0x03E8
DITHER_FREQUENCY	WORD	Dither-Frequenz in [Hz] Wertebereich = 0...FREQUENCY / 2 FREQUENCY / DITHER_FREQUENCY muss geradzahlig sein! Alle anderen Werte erhöht der FB auf den nächst passenden Wert.

6.2.16 Bausteine: Hydraulikregelung

Inhalt

13760

Die Bibliothek `ifm_HYDRAULIC_32bit_Vxxyyzz.Lib` enthält folgende Bausteine:

CONTROL_OCC (→ Seite 336)	OCC = **O**utput **C**urrent **C**ontrol (= stromgeregelter Ausgang) skaliert den Eingangswert [WORD] auf einen angegebenen Strombereich
JOYSTICK_0 (→ Seite 339)	skaliert Signale [INT] aus einem Joystick auf fest definierte Kennlinien, normiert auf 0…1000
JOYSTICK_1 (→ Seite 342)	skaliert Signale [INT] aus einem Joystick auf parametrierbare Kennlinien, normiert auf 0…1000
JOYSTICK_2 (→ Seite 346)	skaliert Signale [INT] aus einem Joystick auf einen parametrierbaren Kennlinien-Verlauf; die Normierung ist frei bestimmbar
NORM_HYDRAULIC (→ Seite 349)	normiert einen Wert [DINT] innerhalb festgelegter Grenzen auf einen Wert mit neuen Grenzen

Aus der Bibliothek `UTIL.Lib` (im CODESYS-Paket) werden folgende Bausteine benötigt:
• RAMP_INT
• CHARCURVE
Diese Bausteine werden von den FBs der Hydraulik-Bibliothek automatisch aufgerufen und parametriert.

Aus der Bibliothek `ifm_CR7132_Vxxyyzz.LIB` werden folgende Bausteine benötigt:

OUTPUT_CURRENT (→ Seite 330)	misst den Strom (Mittelung über Dither-Periode) an einem Ausgangskanal
OUTPUT_CURRENT_CONTROL (→ Seite 331)	Stromregler für einen PWMi-Ausgangskanal

Diese Bausteine werden von den FBs der Hydraulik-Bibliothek automatisch aufgerufen und parametriert.

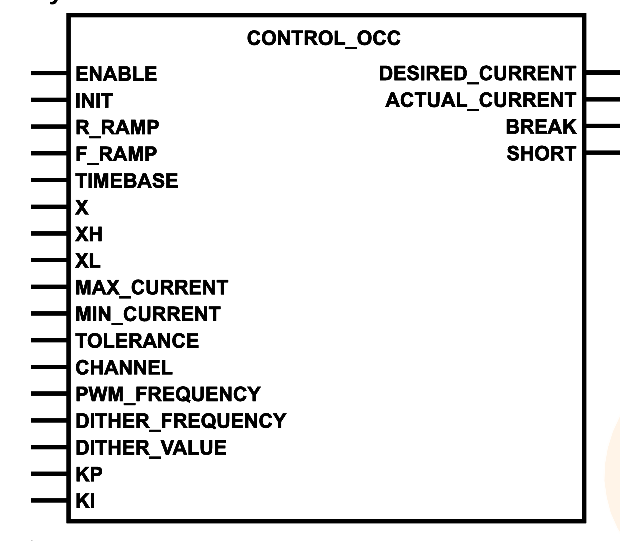
CONTROL_OCC

2735

Baustein-Typ = Funktionsbaustein (FB)

Baustein ist enthalten in Bibliothek `ifm_HYDRAULIC_32bit_Vxxyyzz.Lib`

Symbol in CODESYS:



Beschreibung

2737

CONTROL_OCC skaliert den Eingangswert X auf einen angegebenen Strombereich.

Jede Instanz des FBs wird in jedem SPS-Zyklus einmalig aufgerufen.

Dieser FB nutzt aus der Bibliothek `ifm_CR7132_Vxxyyzz.LIB` folgende FBs:

- ***OUTPUT_CURRENT_CONTROL*** (→ Seite 331)
- ***OUTPUT_CURRENT*** (→ Seite 330)

Der Regler regelt in Abhängigkeit der Periodendauer des PWM Signals.

Die beiden Anstellparameter KI und KP repräsentieren den Integral- und den Proportionalanteil des Reglers. Zur Ermittlung der besten Einstellung des Reglers bietet sich als Startwert an, KI=50 und KP=50 zu setzen.

► Werte für KI und/oder KP vergrößern: ⇨ Regler wird schärfer / schneller
Werte für KI und/oder KP verkleinern: ⇨ Regler wird schwächer / langsamer

> Bei Ausgang DESIRED_CURRENT=0 wird der Ausgang **sofort** auf 0 mA geschaltet, wobei **nicht** entsprechend der eingestellten Parameter auf 0 mA heruntergeregelt wird.

Der Regler verfügt über einen schnellen Ausgleichsmechanismus bei Spannungseinbrüchen der Versorgungsspannung. In Abhängigkeit der Größe des Spannungseinbruchs wird zusätzlich zum Regelverhalten des Reglers die Ratio des PWMs dementsprechend so vergrößert, dass der Regler so schnell wie möglich den Sollwert erreicht.

ℹ Der Eingang X von CONTROL_OCC sollte von einem Ausgang der JOYSTICK-Bausteine gespeist werden.

Parameter der Eingänge

2739

Parameter	Datentyp	Beschreibung
ENABLE	BOOL	TRUE: Baustein ausführen FALSE: Baustein wird nicht ausgeführt > Baustein-Eingänge sind nicht aktiv > Baustein-Ausgänge sind nicht spezifiziert
INIT	BOOL	FALSE ⇨ TRUE (Flanke): Baustein wird initialisiert FALSE: im weiteren Programmablauf
R_RAMP	INT	Steigende Flanke der Rampe in [Inkremente/SPS-Zyklus] 0 = keine Rampe
F_RAMP	INT	Fallende Flanke der Rampe in [Inkremente/SPS-Zyklus] 0 = keine Rampe
TIMEBASE	TIME	Referenz für steigende und fallende Flanke der Rampe: t#0s = steigende / fallende Flanke in [Inkremente/SPS-Zyklus] Schnelle Controller haben sehr kurze Zykluszeiten! sonst = steigende / fallende Flanke in [Inkremente/TIMEBASE]
X	WORD	Eingangswert
XH	WORD	obere Grenze des Eingangswertebereichs [Inkremente]
XL	WORD	untere Grenze des Eingangswertebereichs [Inkremente]
MAX_CURRENT	WORD	Max. Ventilstrom in [mA]
MIN_CURRENT	WORD	Min. Ventilstrom in [mA]
TOLERANCE	BYTE	Toleranz für min. Ventilstrom in [Inkremente] Bei Überschreiten der Toleranz erfolgt Sprung auf MIN_CURRENT
CHANNEL	BYTE	Nummer des stromgeregelten Ausgangskanals (0...15) 0...15 für die Ausgänge Q00...Q15 Für den FB xxx_**E** (falls vorhanden) gilt: 0...15 für die Ausgänge Q00_E...Q15_E
PWM_FREQUENCY	WORD	PWM-Frequenz [Hz] für die Last am Ausgang
DITHER_FREQUENCY	WORD	Dither-Frequenz in [Hz] Wertebereich = 0...FREQUENCY / 2 FREQUENCY / DITHER_FREQUENCY muss geradzahlig sein! Alle anderen Werte erhöht der FB auf den nächst passenden Wert.
DITHER_VALUE	BYTE	Spitze-Spitze-Wert des Dithers in [%] zulässige Werte = 0...100 = 0x00...0x64
KP	BYTE	Proportional-Anteil des Ausgangssignals
KI	BYTE	Integral-Anteil des Ausgangssignals

Für KP, KI gilt: empfohlener Startwert = 50

Parameter der Ausgänge

602

Parameter	Datentyp	Beschreibung
DESIRED_CURRENT	WORD	Stromsollwert in [mA] für OCC (zu Kontrollzwecken)
ACTUAL_CURRENT	WORD	Ausgangsstrom in [mA]
BREAK	BOOL	Fehler: Leitung am Ausgang unterbrochen
SHORT	BOOL	Fehler: Kurzschluss in Leitung am Ausgang

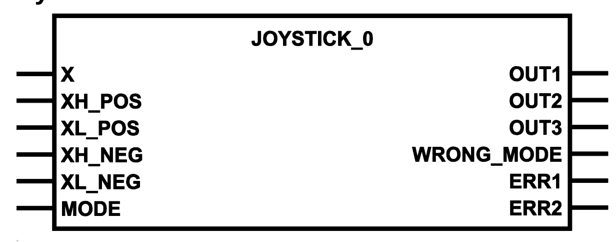
JOYSTICK_0

6250

Baustein-Typ = Funktionsbaustein (FB)

Baustein ist enthalten in Bibliothek `ifm_hydraulic_32bit_Vxxyyzz.Lib`

Symbol in CODESYS:



Beschreibung

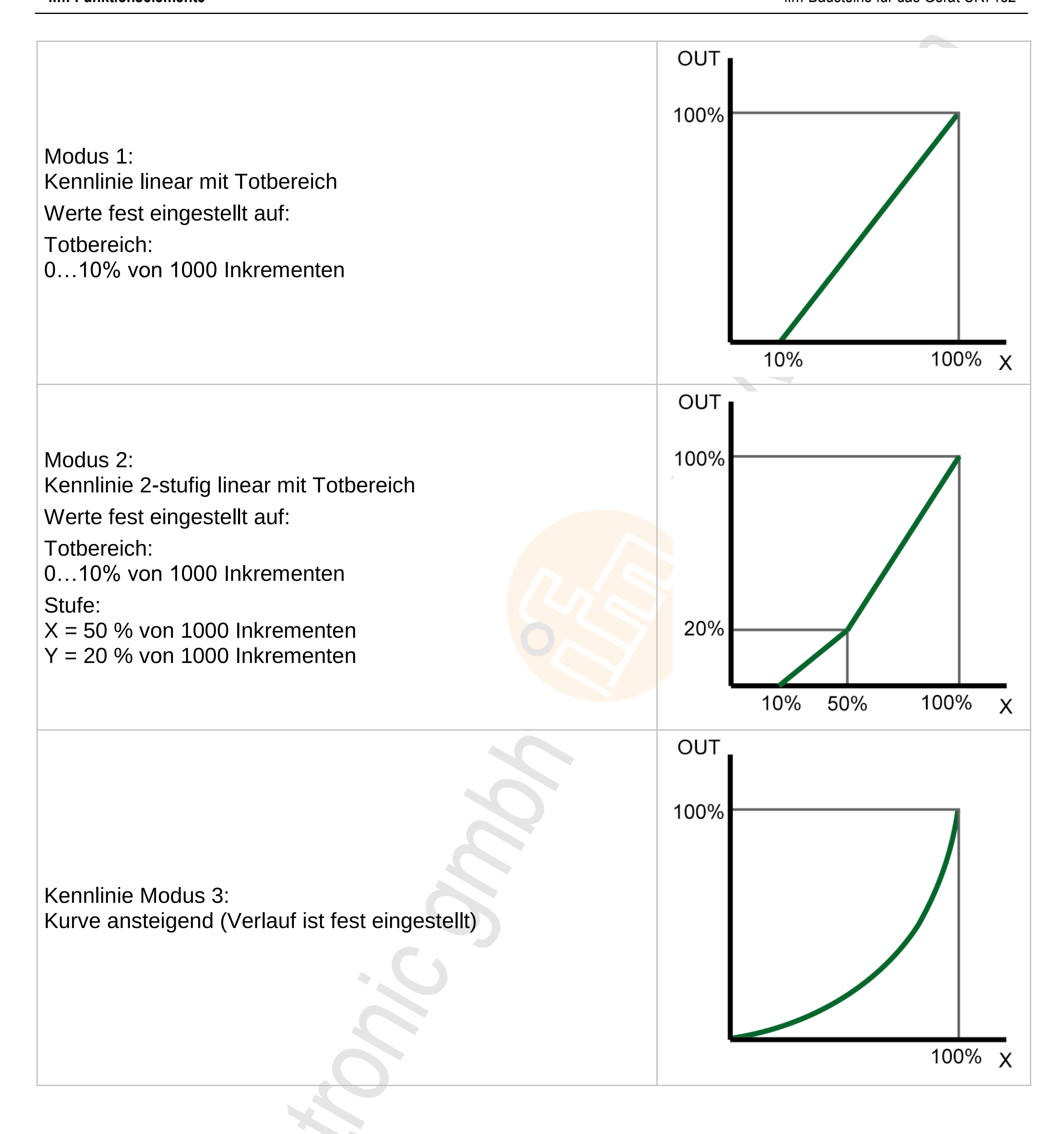
432

JOYSTICK_0 skaliert Signale aus einem Joystick auf fest definierte Kennlinien, normiert auf 0…1000.

Bei diesem FB sind die Kennlinien-Werte fest vorgegeben (→ Grafiken):

- Steigende Flanke der Rampe = 5 Inkremente/SPS-Zyklus
  ⓘ Schnelle Controller haben sehr kurze Zykluszeiten!
- Fallende Flanke der Rampe = keine Rampe

Die Parameter XL_POS (XL+), XH_POS (XH+), XL_NEG (XL-) und XH_NEG (XH-) dienen dazu, die Joystickbewegung nur im erwünschten Bewegungsbereich auszuwerten. Die Werte für den positiven und den negativen Bereich dürfen sich unterscheiden. Die Werte für XL_NEG und XH_NEG sind hier negativ.	XH- XL- XL+ XH+
Modus 0: Kennlinie linear für den Bereich XL bis XH	OUT 100% 100% X



Modus 1: Kennlinie linear mit Totbereich Werte fest eingestellt auf: Totbereich: 0…10% von 1000 Inkrementen	OUT, 100%, 10%, 100%, X
Modus 2: Kennlinie 2-stufig linear mit Totbereich Werte fest eingestellt auf: Totbereich: 0…10% von 1000 Inkrementen Stufe: X = 50 % von 1000 Inkrementen Y = 20 % von 1000 Inkrementen	OUT, 100%, 20%, 10%, 50%, 100%, X
Kennlinie Modus 3: Kurve ansteigend (Verlauf ist fest eingestellt)	OUT, 100%, 100%, X

Parameter der Eingänge

433

Parameter	Datentyp	Beschreibung
X	INT	Eingangswert [Inkremente]
XH_POS	INT	Max. Sollwert positive Richtung [Inkremente] (auch negative Werte zulässig)
XL_POS	INT	Min. Sollwert positive Richtung [Inkremente] (auch negative Werte zulässig)
XH_NEG	INT	Max. Sollwert negative Richtung [Inkremente] (auch negative Werte zulässig)
XL_NEG	INT	Min. Sollwert negative Richtung [Inkremente] (auch negative Werte zulässig)
MODE	BYTE	Modus Auswahl Kennlinie: 0 = linear (X|OUT = 0|0 ... 1000|1000) 1 = linear mit Totbereich (X|OUT = 0|0 ... 100|0 ... 1000|1000) 2 = 2-stufig linear mit Totbereich (X|OUT = 0|0 ... 100|0 ... 500|200 ... 1000|1000) 3 = Kurve ansteigend (Verlauf ist fest eingestellt)

Parameter der Ausgänge

6252

Parameter	Datentyp	Beschreibung
OUT1	WORD	normierter Ausgangswert: 0…1000 Inkremente z.B. für Ventil links
OUT2	WORD	normierter Ausgangswert: 0…1000 Inkremente z.B. für Ventil rechts
OUT3	INT	normierter Ausgangswert: -1000…0…1000 Inkremente z.B. für Ventil an Ausgangsmodul (z.B. CR2011 oder CR2031)
WRONG_MODE	BOOL	Fehler: Ungültiger Modus
ERR1	BYTE	Fehler-Code für steigende Flanke (bezogen auf die intern verwendeten FBs CHARCURVE und RAMP_INT aus der `util.lib`) (mögliche Meldungen → folgende Tabelle)
ERR2	BYTE	Fehler-Code für fallende Flanke (bezogen auf die intern verwendeten FBs CHARCURVE und RAMP_INT aus der `util.lib`) (mögliche Meldungen → folgende Tabelle)

Mögliche Ergebnisse für ERR1 und ERR2:

Wert dez	hex	Beschreibung
0	00	kein Fehler
1	01	Fehler in Zahlenreihe: Falsche Reihenfolge
2	02	Fehler: Eingangswert IN ist nicht im Wertebereich der Zahlenreihe
4	04	Fehler: Ungültige Anzahl N für Zahlenreihe

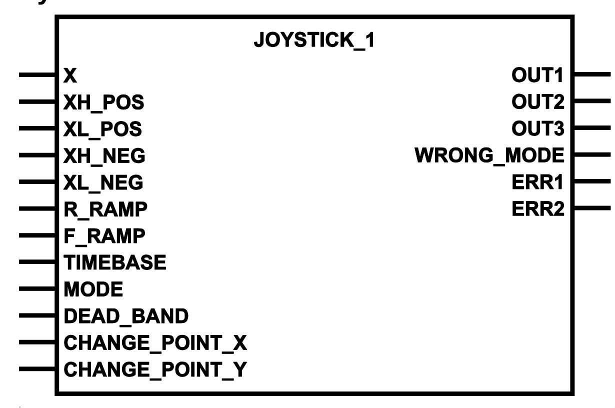
JOYSTICK_1

6255

Baustein-Typ = Funktionsbaustein (FB)

Baustein ist enthalten in Bibliothek `ifm_hydraulic_32bit_Vxxyyzz.Lib`

Symbol in CODESYS:

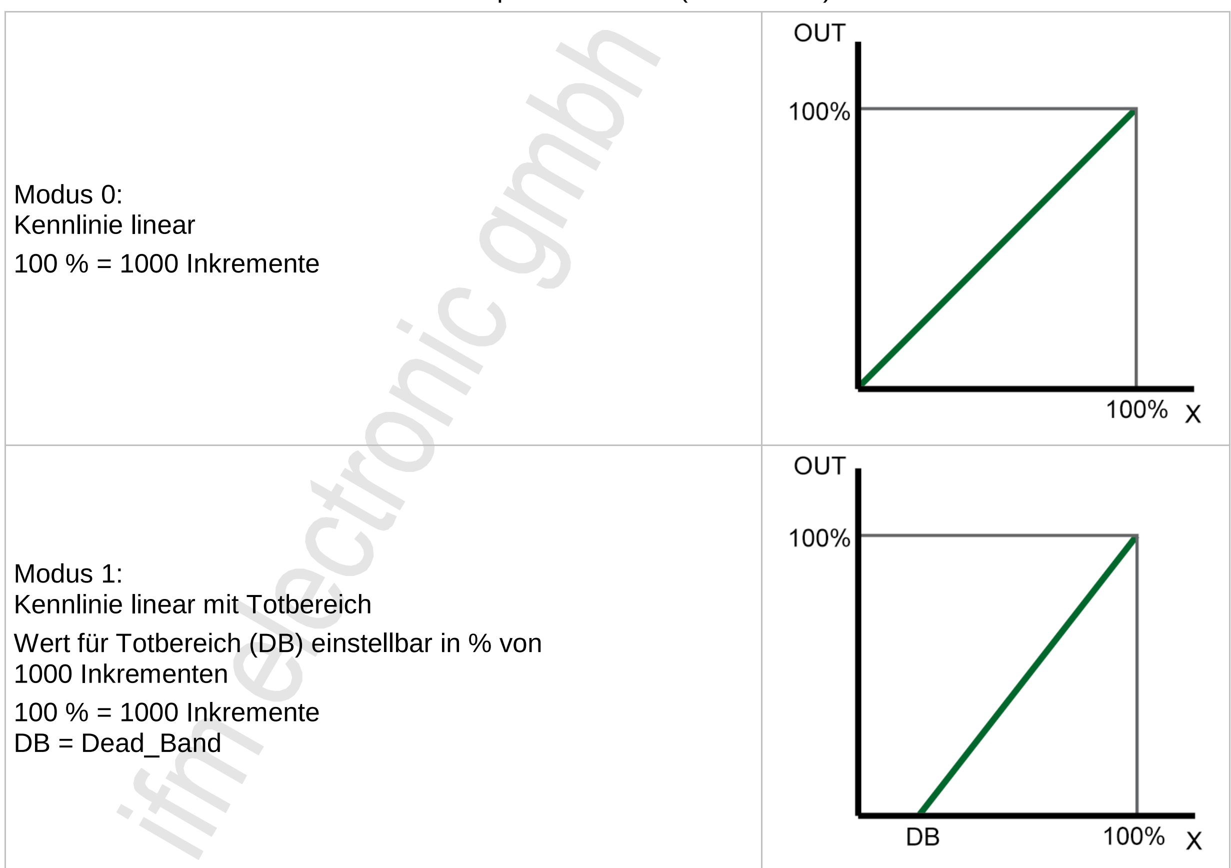


Beschreibung

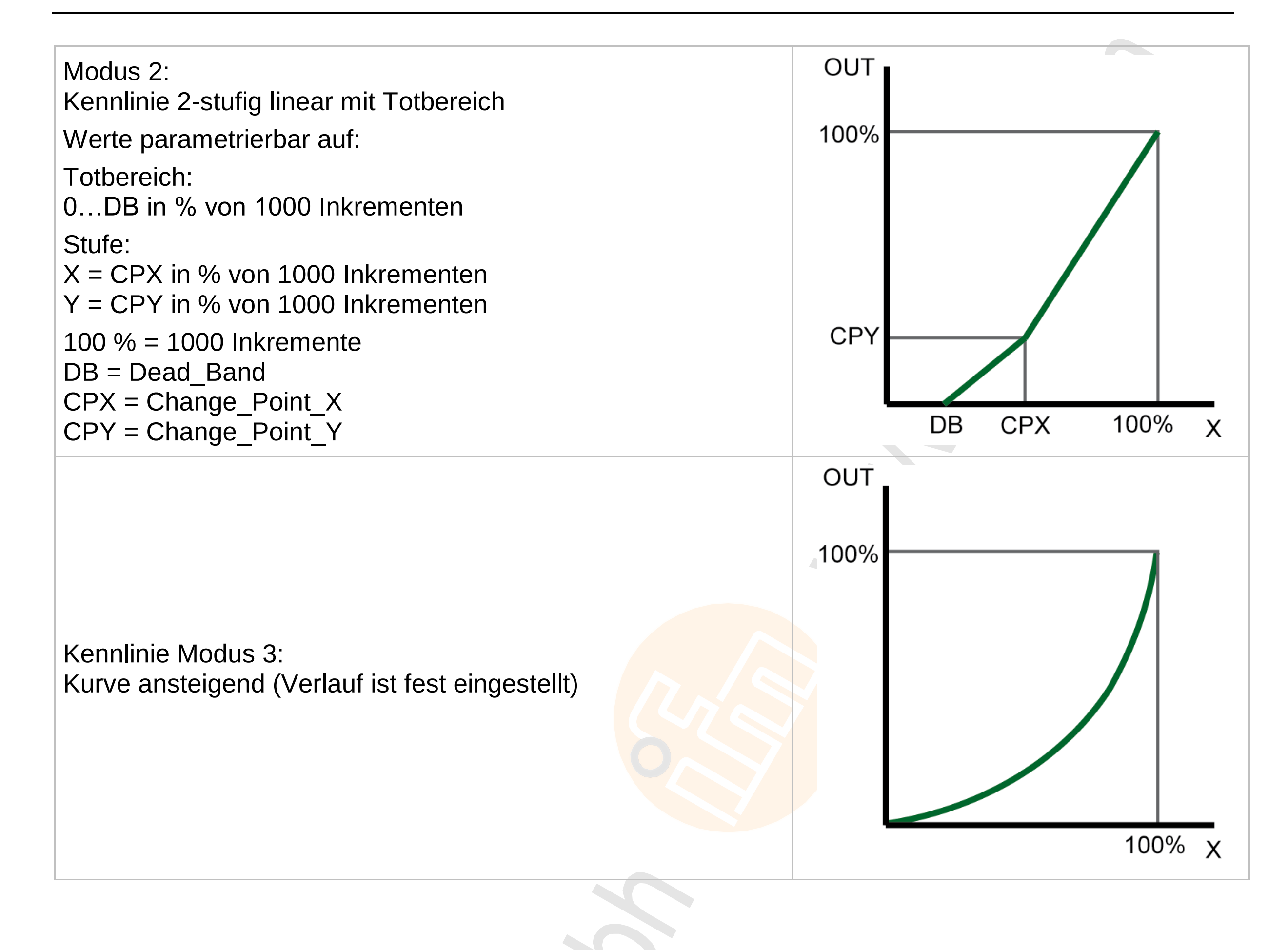
425

JOYSTICK_1 skaliert Signale aus einem Joystick auf parametrierbare Kennlinien, normiert auf 0…1000.

Bei diesem FB sind die Kennlinien-Werte parametrierbar (→ Grafiken):



Modus 0: Kennlinie linear 100 % = 1000 Inkremente	
Modus 1: Kennlinie linear mit Totbereich Wert für Totbereich (DB) einstellbar in % von 1000 Inkrementen 100 % = 1000 Inkremente DB = Dead_Band	



Modus 2: Kennlinie 2-stufig linear mit Totbereich Werte parametrierbar auf: Totbereich: 0…DB in % von 1000 Inkrementen Stufe: X = CPX in % von 1000 Inkrementen Y = CPY in % von 1000 Inkrementen 100 % = 1000 Inkremente DB = Dead_Band CPX = Change_Point_X CPY = Change_Point_Y	OUT, 100%, CPY, DB, CPX, 100%, X
Kennlinie Modus 3: Kurve ansteigend (Verlauf ist fest eingestellt)	OUT, 100%, 100%, X

Parameter der Eingänge

6256

Parameter	Datentyp	Beschreibung
X	INT	Eingangswert [Inkremente]
XH_POS	INT	Max. Sollwert positive Richtung [Inkremente] (auch negative Werte zulässig)
XL_POS	INT	Min. Sollwert positive Richtung [Inkremente] (auch negative Werte zulässig)
XH_NEG	INT	Max. Sollwert negative Richtung [Inkremente] (auch negative Werte zulässig)
XL_NEG	INT	Min. Sollwert negative Richtung [Inkremente] (auch negative Werte zulässig)
R_RAMP	INT	Steigende Flanke der Rampe in [Inkremente/SPS-Zyklus] 0 = keine Rampe
F_RAMP	INT	Fallende Flanke der Rampe in [Inkremente/SPS-Zyklus] 0 = keine Rampe
TIMEBASE	TIME	Referenz für steigende und fallende Flanke der Rampe: t#0s = steigende / fallende Flanke in [Inkremente/SPS-Zyklus] ! Schnelle Controller haben sehr kurze Zykluszeiten! sonst = steigende / fallende Flanke in [Inkremente/TIMEBASE]
MODE	BYTE	Modus Auswahl Kennlinie: 0 = linear (X|OUT = 0|0 ... 1000|1000) 1 = linear mit Totbereich (X|OUT = 0|0 ... DB|0 ... 1000|1000) 2 = 2-stufig linear mit Totbereich (X|OUT = 0|0 ... DB|0 ... CPX|CPY ... 1000|1000) 3 = Kurve ansteigend (Verlauf ist fest eingestellt)
DEAD_BAND	BYTE	Einstellbarer Totbereich in [% von 1000 Inkrementen]
CHANGE_POINT_X	BYTE	Für Modus 2: Rampenstufe, Wert für X in [% von 1000 Inkrementen]
CHANGE_POINT_Y	BYTE	Für Modus 2: Rampenstufe, Wert für Y in [% von 1000 Inkrementen]

Parameter der Ausgänge

6252

Parameter	Datentyp	Beschreibung
OUT1	WORD	normierter Ausgangswert: 0…1000 Inkremente z.B. für Ventil links
OUT2	WORD	normierter Ausgangswert: 0…1000 Inkremente z.B. für Ventil rechts
OUT3	INT	normierter Ausgangswert: -1000…0…1000 Inkremente z.B. für Ventil an Ausgangsmodul (z.B. CR2011 oder CR2031)
WRONG_MODE	BOOL	Fehler: Ungültiger Modus
ERR1	BYTE	Fehler-Code für steigende Flanke (bezogen auf die intern verwendeten FBs CHARCURVE und RAMP_INT aus der `util.lib`) (mögliche Meldungen → folgende Tabelle)
ERR2	BYTE	Fehler-Code für fallende Flanke (bezogen auf die intern verwendeten FBs CHARCURVE und RAMP_INT aus der `util.lib`) (mögliche Meldungen → folgende Tabelle)

Mögliche Ergebnisse für ERR1 und ERR2:

Wert dez	hex	Beschreibung
0	00	kein Fehler
1	01	Fehler in Zahlenreihe: Falsche Reihenfolge
2	02	Fehler: Eingangswert IN ist nicht im Wertebereich der Zahlenreihe
4	04	Fehler: Ungültige Anzahl N für Zahlenreihe

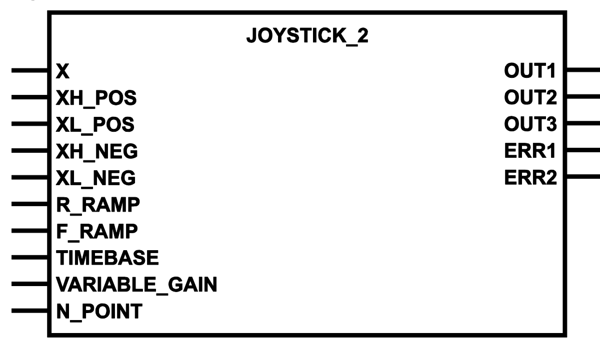
JOYSTICK_2

6258

Baustein-Typ = Funktionsbaustein (FB)

Baustein ist enthalten in Bibliothek `ifm_hydraulic_32bit_Vxxyyzz.Lib`

Symbol in CODESYS:

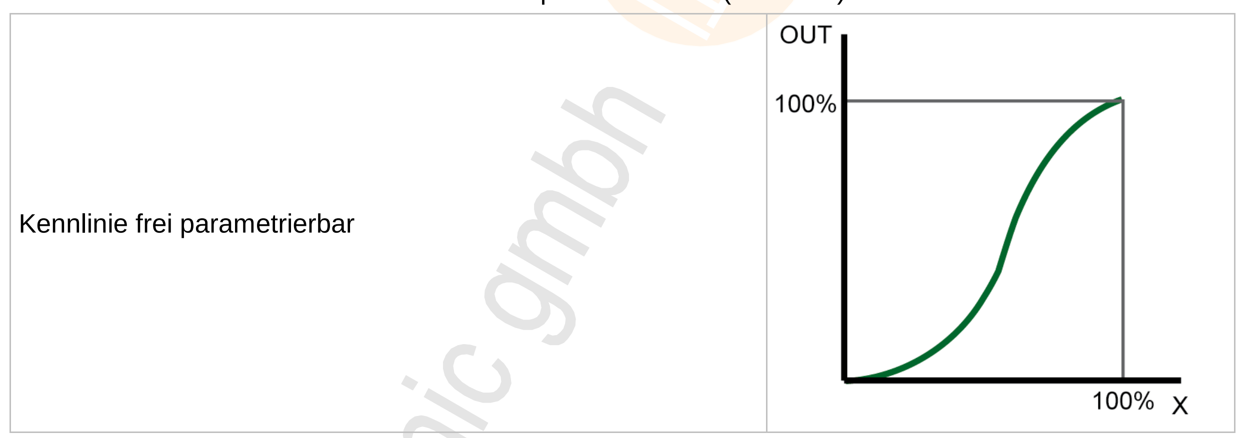


Beschreibung

418

JOYSTICK_2 skaliert Signale aus einem Joystick auf einen parametrierbaren Kennlinien-Verlauf. Die Normierung ist frei bestimmbar.

Bei diesem FB ist der Kennlinien-Verlauf frei parametrierbar (→ Grafik):



Parameter der Eingänge

6261

Parameter	Datentyp	Beschreibung
X	INT	Eingangswert [Inkremente]
XH_POS	INT	Max. Sollwert positive Richtung [Inkremente] (auch negative Werte zulässig)
XL_POS	INT	Min. Sollwert positive Richtung [Inkremente] (auch negative Werte zulässig)
XH_NEG	INT	Max. Sollwert negative Richtung [Inkremente] (auch negative Werte zulässig)
XL_NEG	INT	Min. Sollwert negative Richtung [Inkremente] (auch negative Werte zulässig)
R_RAMP	INT	Steigende Flanke der Rampe in [Inkremente/SPS-Zyklus] 0 = keine Rampe
F_RAMP	INT	Fallende Flanke der Rampe in [Inkremente/SPS-Zyklus] 0 = keine Rampe
TIMEBASE	TIME	Referenz für steigende und fallende Flanke der Rampe: t#0s = steigende / fallende Flanke in [Inkremente/SPS-Zyklus] ! Schnelle Controller haben sehr kurze Zykluszeiten! sonst = steigende / fallende Flanke in [Inkremente/TIMEBASE]

VARIABLE_GAIN	ARRAY [0..10] OF POINT	Wertepaare, die den Kurven-Verlauf beschreiben
		Es werden die ersten in N_POINT angegebenen Wertepaare verwertet. n = 2…11
		Beispiel: 9 Wertepaare als Variable VALUES deklariert:
		`VALUES : ARRAY [0..10] OF POINT :=`
		`(X:=0,Y:=0),(X:=200,Y:=0), (X:=300,Y:=50),`
		`(X:=400,Y:=100), (X:=700,Y:=500),`
		`(X:=1000,Y:=900), (X:=1100,Y:=950),`
		`(X:=1200,Y:=1000), (X:=1400,Y:=1050);`
		Zwischen den Werten dürfen auch Leerzeichen stehen.

N_POINT	BYTE	Anzahl der Punkte (Wertepaare in VARIABLE_GAIN), womit die Kurven-Charakteristik definiert ist: n = 2…11

Parameter der Ausgänge

420

Parameter	Datentyp	Beschreibung
OUT1	WORD	normierter Ausgangswert: 0…1000 Inkremente z.B. für Ventil links
OUT2	WORD	normierter Ausgangswert: 0…1000 Inkremente z.B. für Ventil rechts
OUT3	INT	normierter Ausgangswert: -1000…0…1000 Inkremente z.B. für Ventil an Ausgangsmodul (z.B. CR2011 oder CR2031)
ERR1	BYTE	Fehler-Code für steigende Flanke (bezogen auf die intern verwendeten FBs CHARCURVE und RAMP_INT aus der `util.lib`) (mögliche Meldungen → folgende Tabelle)
ERR2	BYTE	Fehler-Code für fallende Flanke (bezogen auf die intern verwendeten FBs CHARCURVE und RAMP_INT aus der `util.lib`) (mögliche Meldungen → folgende Tabelle)

Mögliche Ergebnisse für ERR1 und ERR2:

Wert dez	hex	Beschreibung
0	00	kein Fehler
1	01	Fehler in Zahlenreihe: Falsche Reihenfolge
2	02	Fehler: Eingangswert IN ist nicht im Wertebereich der Zahlenreihe
4	04	Fehler: Ungültige Anzahl N für Zahlenreihe

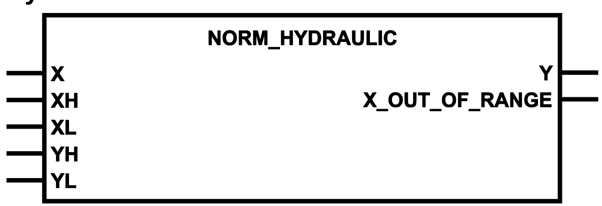
NORM_HYDRAULIC

394

Baustein-Typ = Funktionsbaustein (FB)

Baustein ist enthalten in Bibliothek `ifm_hydraulic_32bit_Vxxyyzz.Lib`

Symbol in CODESYS:



Beschreibung

397

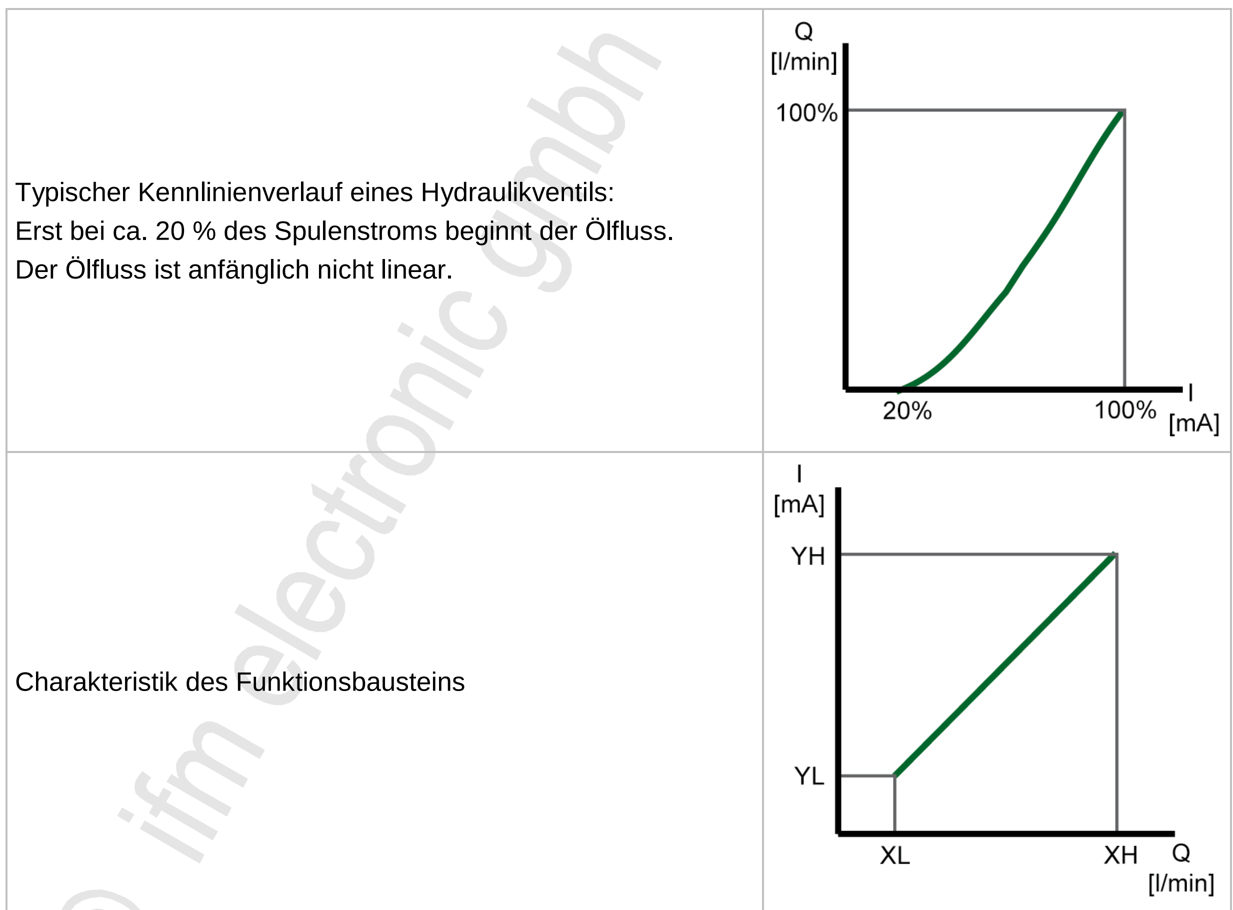
NORM_HYDRAULIC normiert Eingangswerte innerhalb festgesetzter Grenzen auf Werte mit neuen Grenzen.

ℹ Dieser FB entspricht NORM_DINT aus der CODESYS-Bibliothek `UTIL.Lib`.

Der FB normiert einen Wert vom Typ DINT, der innerhalb der Grenzen zwischen XH und XL liegt, auf einen Ausgangswert innerhalb der Grenzen zwischen YH und YL.

Bedingt durch Rundungsfehler können Abweichungen beim normierten Wert um 1 auftreten. Werden die Grenzen (XH/XL oder YH/YL) invertiert angegeben, erfolgt auch die Normierung invertiert.

Wenn X außerhalb der Grenzen XL…XH liegt, wird die Fehlermeldung X_OUT_OF_RANGE = TRUE.



Typischer Kennlinienverlauf eines Hydraulikventils: Erst bei ca. 20 % des Spulenstroms beginnt der Ölfluss. Der Ölfluss ist anfänglich nicht linear.	
Charakteristik des Funktionsbausteins	

Parameter der Eingänge

398

Parameter	Datentyp	Beschreibung
X	DINT	Eingangswert
XH	DINT	Max. Eingangswert [Inkremente]
XL	DINT	Min. Eingangswert [Inkremente]
YH	DINT	Max. Ausgangswert [Inkremente], z.B.: Ventilstrom [mA], Durchfluss [l/min]
YL	DINT	Min. Ausgangswert [Inkremente], z.B.: Ventilstrom [mA], Durchfluss [l/min]

Parameter der Ausgänge

399

Parameter	Datentyp	Beschreibung
Y	DINT	Ausgangswert
X_OUT_OF_RANGE	BOOL	Fehler: X liegt außerhalb der Grenzen von XH und XL

Beispiel: NORM_HYDRAULIC

400

Parameter	Fall 1	Fall 2	Fall 3
oberer Grenzwert Eingang XH	100	100	2000
unterer Grenzwert Eingang XL	0	0	0
oberer Grenzwert Ausgang YH	2000	0	100
unterer Grenzwert Ausgang YL	0	2000	0
nicht normierter Wert X	20	20	20
normierter Wert Y	400	1600	1

- Fall 1:
  Eingang mit relativ grober Auflösung.
  Ausgang mit hoher Auflösung.
  1 X-Inkrement ergibt 20 Y-Inkremente.
- Fall 2:
  Eingang mit relativ grober Auflösung.
  Ausgang mit hoher Auflösung.
  1 X-Inkrement ergibt 20 Y-Inkremente.
  Ausgangssignal ist gegenüber dem Eingangssignal invertiert.
- Fall 3:
  Eingang mit hoher Auflösung.
  Ausgang mit relativ grober Auflösung.
  20 X-Inkremente ergeben 1 Y-Inkrement.

6.2.17 Bausteine: Regler

Inhalt

1634

Der nachfolgende Abschnitt beschreibt im Detail die Bausteine, die zum Aufbau von Software-Reglern im **ecomat*mobile***-Gerät bereitgestellt werden. Die Bausteine können auch als Basis für die Entwicklung von eigenen Regelungsfunktionen genutzt werden.

Einstellregel für einen Regler

1627

Für Regelstrecken, deren Zeitkonstanten nicht bekannt sind, ist das Einstellverfahren nach Ziegler und Nickols im geschlossenen Regelkreis vorteilhaft:

Einstellregel

1628

Die Regeleinrichtung wird zunächst als eine reine P-Regeleinrichtung betrieben. Dazu wird die Vorhaltezeit T_V auf 0 und die Nachstellzeit T_N auf einen sehr großen Wert (ideal auf unendlich) für eine träge Strecke eingestellt. Bei einer schnellen Regelstrecke sollte ein kleines T_N gewählt werden.

Der Proportionalbeiwert KP wird anschließend solange vergrößert, bis die Regel- und die Stellabweichung bei KP = $KP_{kritisch}$ Dauerschwingungen mit konstanter Amplitude ausführen. Es ist damit die Stabilitätsgrenze erreicht.

Anschließend muss die Periodendauer $T_{kritisch}$ der Dauerschwingung ermittelt werden.

Nur bei Bedarf einen D-Anteil hinzufügen.

T_V sollte ca. 2...10-mal kleiner sein als T_N.

KP sollte gleich groß wie KD gewählt werden.

Idealisiert ist die Regelstrecke wie folgt einzustellen:

Regeleinrichtung	KP = KD	TN	TV
P	2,0 • $KP_{kritisch}$	—	—
PI	2,2 • $KP_{kritisch}$	0,83 • $T_{kritisch}$	—
PID	1,7 • $KP_{kritisch}$	0,50 • $T_{kritisch}$	0,125 • $T_{kritisch}$

(!) Bei diesem Einstellverfahren darauf achten, dass die Regelstrecke durch die auftretenden Schwingungen keinen Schaden nimmt. Bei empfindlichen Regelstrecken darf KP nur bis zu einem Wert erhöht werden, bei dem sicher noch keine Schwingungen auftreten.

Dämpfung von Überschwingungen

1629

Um Überschwingungen zu dämpfen, kann ***PT1*** (→ Seite 356) (Tiefpass) eingesetzt werden. Dazu wird der Sollwert XS durch das PT1-Glied gedämpft, bevor er der Reglerfunktion zugeführt wird.

Die Einstellgröße T1 sollte ca. 4...5-mal größer sein als TN des Reglers.

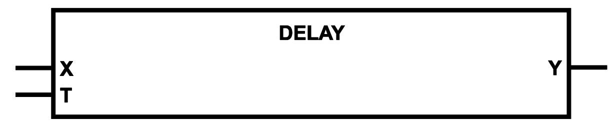
DELAY

585

Baustein-Typ = Funktionsbaustein (FB)

Baustein ist enthalten in Bibliothek `ifm_CR7132_Vxxyyzz.LIB`

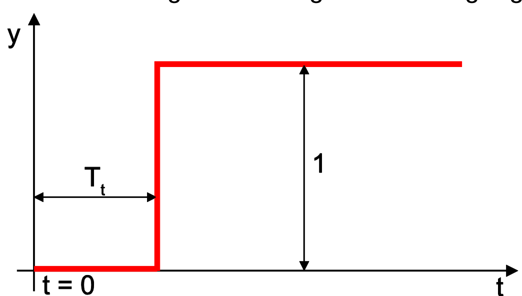
Symbol in CODESYS:



Beschreibung

588

DELAY verzögert die Ausgabe des Eingangswertes um die Zeit T (Totzeit-Glied).



Grafik: Zeitlicher Verlauf von DELAY

Die Totzeit wird durch die Dauer des SPS-Zyklus beeinflusst.
Die Totzeit darf nicht länger sein als 100 • SPS-Zykluszeit (Speichergrenze!).
Wird eine größere Verzögerung eingestellt, wird die Auflösung der Werte am Ausgang des FB schlechter, wodurch kurze Werteänderungen verloren gehen können.

> Damit der FB einwandfrei arbeitet: FB in jedem SPS-Zyklus aufrufen!

Parameter der Eingänge

2615

Parameter	Datentyp	Beschreibung
X	REAL	Eingangswert
T	TIME	Verzögerungszeit (Totzeit) zulässig: 0...100 • Zykluszeit

Parameter der Ausgänge

2616

Parameter	Datentyp	Beschreibung
Y	REAL	Eingangswert, verzögert um die Zeit T

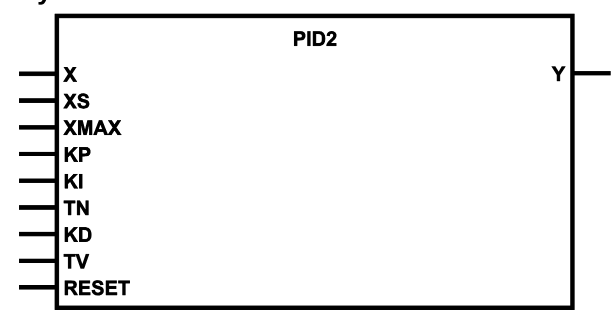
PID2

344

Baustein-Typ = Funktionsbaustein (FB)

Baustein ist enthalten in Bibliothek `ifm_CR7132_Vxxyyzz.LIB`

Symbol in CODESYS:



Beschreibung

6262

PID2 organisiert einen PID-Regler.

Die Änderung der Stellgröße eines PID-Reglers setzt sich aus einem proportionalen, integralen und differentialen Anteil zusammen. Die Stellgröße ändert sich zunächst um einen von der Änderungsgeschwindigkeit der Eingangsgröße abhängigen Betrag (**D**ifferential-Anteil). Nach Ablauf der Vorhaltezeit TV geht die Stellgröße auf den dem Proportionalbereich entsprechenden Wert zurück und ändert sich dann entsprechend der Nachstellzeit TN.

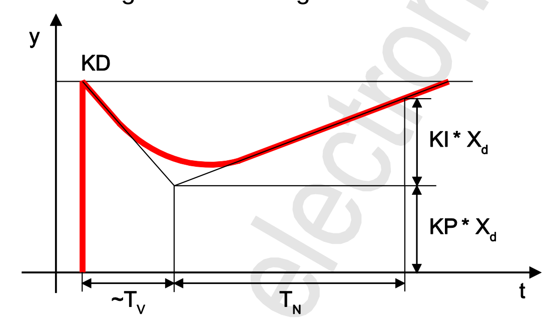
Die Stellgröße Y ist bereits auf ***PWM1000*** (→ Seite 333) normiert.

Regeln:

- Negative Werte bei KP, KI und KD sind nicht zulässig.
- Bei TN = 0 wird der I-Anteil nicht berechnet.
- Bei XS > XMAX wird XS auf XMAX limitiert.
- Bei X > XMAX wird Y auf 0 gesetzt.
- Wenn X > XS, dann wird die Stellgröße erhöht.
- Wenn X < XS, dann wird die Stellgröße reduziert.

Eine Führungsgröße wird intern zur Stellgröße hinzuaddiert:
Y = Y + 65 536 - (XS / XMAX • 65 536).

Die Stellgröße Y hat folgenden zeitlichen Verlauf.



Grafik: Typische Sprungantwort eines PID-Reglers

Parameter der Eingänge

12963

Parameter	Datentyp	Beschreibung
X	WORD	Eingangswert
XS	WORD	Sollwert
XMAX	WORD	Maximaler Istwert zur Festlegung des Istwert-Wertebereichs
KP	REAL	Proportional-Anteil des Ausgangssignals
KI	REAL	Integral-Anteil des Ausgangssignals
TN	TIME	Nachstellzeit (Integral-Anteil)
KD	REAL	Differential-Anteil des Ausgangssignals
TV	TIME	Vorhaltezeit (Differential-Anteil)
RESET	BOOL	TRUE: Regler zurücksetzen FALSE: Funktion wird nicht ausgeführt

Parameter der Ausgänge

349

Parameter	Datentyp	Beschreibung
Y	WORD	Stellgröße (0...1000 ‰)

Einstellempfehlung

350

- ► TN gemäß des Zeitverhaltens der Strecke wählen
(schnelle Strecke = kleines TN, träge Strecke = großes TN)
- ► KP langsam, schrittweise erhöhen bis zu einem Wert, bei dem sicher noch kein Schwingen auftritt.
- ► TN bei Bedarf nachjustieren
- ► Nur bei Bedarf D-Anteil hinzufügen:
TV ca. 2...10-mal kleiner als TN wählen.
KD etwa gleich groß wie KP wählen.

Beachten Sie, dass die maximale Regelabweichung + 127 beträgt. Für ein gutes Regelverhalten sollte dieser Bereich einerseits nicht überschritten, andererseits aber möglichst ausgenutzt werden.

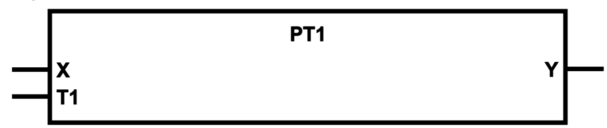
PT1

338

Baustein-Typ = Funktionsbaustein (FB)

Baustein ist enthalten in Bibliothek `ifm_CR7132_Vxxyyzz.LIB`

Symbol in CODESYS:



Beschreibung

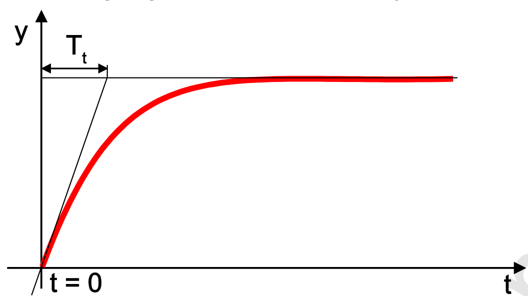
341

PT1 organisiert eine Regelstrecke mit Verzögerung 1. Ordnung.

Bei der Funktion handelt es sich um eine proportionale Regelstrecke mit Verzögerung. Sie wird z.B. zur Bildung von Rampen bei Einsatz der PWM-Funktionen genutzt.

! Der Ausgang des FB kann instabil werden, wenn T1 kleiner ist als die SPS-Zykluszeit.

Die Ausgangsvariable Y des Tiefpassfilters hat folgenden zeitlichen Verlauf (Einheitssprungfunktion):



Grafik: Zeitlicher Verlauf bei PT1

Parameter der Eingänge

2618

Parameter	Datentyp	Beschreibung
X	DINT	Eingangswert
T1	TIME	Verzögerungszeit (Zeitkonstante)

Parameter der Ausgänge

2619

Parameter	Datentyp	Beschreibung
Y	DINT	Ausgangswert

6.2.18 Bausteine: Zeit messen / setzen

Inhalt

1601

Mit folgenden Bausteinen der **ifm electronic** können Sie...

- Zeiten messen und im Anwendungsprogramm auswerten,
- bei Bedarf Zeitwerte ändern.

TIMER_READ

236

Baustein-Typ = Funktionsbaustein (FB)

Baustein ist enthalten in Bibliothek `ifm_CR7132_Vxxyyzz.LIB`

Symbol in CODESYS:



Beschreibung

239

TIMER_READ liest die aktuelle Systemzeit aus.

Mit Anlegen der Versorgungsspannung bildet das Gerät einen Zeittakt, der in einem Register aufwärts gezählt wird. Dieses Register kann mittels des Funktionsaufrufes ausgelesen und z.B. zur Zeitmessung genutzt werden.

> Der System-Timer läuft maximal bis 0xFFFF FFFF (entspricht 49d 17h 2min 47s 295ms) und startet anschließend wieder mit 0.

Parameter der Ausgänge

241

Parameter	Datentyp	Beschreibung
T	TIME	Aktuelle Systemzeit [ms]

TIMER_READ_US

657

Baustein-Typ = Funktionsbaustein (FB)

Baustein ist enthalten in Bibliothek `ifm_CR7132_Vxxyyzz.LIB`

Symbol in CODESYS:



Beschreibung

660

TIMER_READ_US liest die aktuelle Systemzeit in [µs] aus.

Mit Anlegen der Versorgungsspannung bildet das Gerät einen Zeittakt, der in einem Register aufwärts gezählt wird. Dieses Register kann mittels des FB-Aufrufes ausgelesen werden und z.B. zur Zeitmessung genutzt werden.

> **Info**
>
> Der System-Timer läuft maximal bis zum Zählerwert 1h 11min 34s 967ms 295µs und startet anschließend wieder mit 0.

Parameter der Ausgänge

662

Parameter	Datentyp	Beschreibung
TIME_US	DWORD	Aktuelle Systemzeit [µs]

6.2.19 Bausteine: Gerätetemperatur auslesen

Inhalt

2364

Mit folgendem Baustein zeigt Ihnen das Gerät die Innentemperatur.

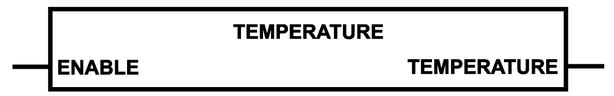
TEMPERATURE

2216

Baustein-Typ = Funktionsbaustein (FB)

Baustein ist enthalten in Bibliothek `ifm_CR7132_Vxxyyzz.LIB`

Symbol in CODESYS:



Beschreibung

2365

TEMPERATURE liest die aktuelle Temperatur im Gerät aus.
Der FB kann zyklisch aufgerufen werden und zeigt am Ausgang die aktuelle Gerätetemperatur an.

Parameter der Eingänge

2366

Parameter	Datentyp	Beschreibung
ENABLE	BOOL	TRUE: Baustein ausführen FALSE: Baustein wird nicht ausgeführt > Baustein-Eingänge sind nicht aktiv > Baustein-Ausgänge sind nicht spezifiziert

Parameter der Ausgänge

2367

Parameter	Datentyp	Beschreibung
TEMPERATURE	INT	Aktuelle Geräteinnentemperatur [°C]

6.2.20 Bausteine: Daten im Speicher sichern, lesen und wandeln

Inhalt

13795

Speicherarten zur Datensicherung

13805

Das Gerät bietet folgende Speicher:

Flash-Speicher

13803

Eigenschaften:

- schnelles Schreiben und Lesen
- begrenzte Schreib-/Lesehäufigkeit
- nur zum Speichern großer Datenmengen sinnvoll einsetzbar
- vor dem erneuten Schreiben muss Speicherinhalt gelöscht werden
- Daten sichern mit FLASHWRITE
- Daten lesen mit FLASHREAD

FRAM-Speicher

13802

FRAM steht hier allgemein für alle Arten von nichtflüchtigen, schnellen Speichern.

Eigenschaften:

- schnelles Schreiben und Lesen
- unbegrenzte Schreib-/Lesehäufigkeit
- beliebige Speicherbereiche wählbar
- Daten sichern mit FRAMWRITE
- Daten lesen mit FRAMREAD

Dateisystem

2690

Das Dateisystem koordiniert, wo im Speicher welche Informationen liegen. Die Größe des Dateisystems beträgt 128 kByte.

Die Dateinamen des Dateisystems sind begrenzt:
max. Länge für Controller: CR0n3n, CR7n3n: 15 Zeichen
max. Länge für alle anderen Geräte: 11 Zeichen

Verhalten des Dateisystems im Controller: CR0n3n, CR7n3n**:**

- Der Controller versucht immer, die Datei zu schreiben, auch wenn der gleiche Dateiname bereits existiert. Gegebenenfalls wird die Datei mehrfach gespeichert.Genutzt wird nur die aktuelle Datei. Über den Download (s.u.) wird diese Mehrfach-Ablage vermieden.
- Einzelne Dateien können nicht überschrieben oder gelöscht werden.
- Das Dateisystem wird bei jedem Download (Bootprojekt-Download oder RAM-Download) komplett gelöscht. Anschließend kann z.B. eine Symboldatei oder eine Projektdatei (Funktionen in CODESYS) geschrieben werden.
- Das Dateisystem wird ebenfalls bei einem [Reset (Ursprung)] (CODESYS-Funktion im Menü [Online]) gelöscht.

Automatische Datensicherung

Inhalt

14168
2347

Die **ecomat*mobile***-Geräte bieten die Möglichkeit, Daten (BOOL, BYTE, WORD, DWORD) remanent (= spannungsausfallsicher) im Speicher zu sichern. Voraussetzung ist, dass die Daten als RETAIN-Variablen angelegt wurden (→ CODESYS).

Man unterscheidet zwischen Variablen, die als RETAIN deklariert wurden, und Variablen im Merkerbereich, der als Block mit ***MEMORY_RETAIN_PARAM*** (→ Seite 365) als remanent konfiguriert werden kann.

Details → Kapitel ***Variablen*** (→ Seite 188)

Der Vorteil des automatischen Speicherns ist, dass auch bei einem plötzlichen Spannungsabfall oder einer Unterbrechung der Versorgungsspannung die aktuellen Werte der Daten erhalten bleiben (z.B. Zählerstände).

> Wenn Versorgungsspannung < 8 V, werden keine Retain-Daten mehr gesichert!
> In diesem Fall wird Merker RETAIN_WARNING = TRUE.

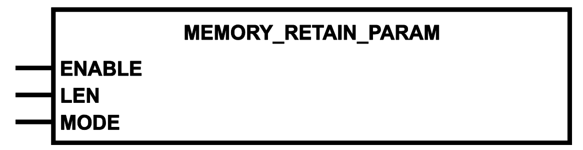
MEMORY_RETAIN_PARAM

2372

Baustein-Typ = Funktionsbaustein (FB)

Baustein ist enthalten in Bibliothek `ifm_CR7132_Vxxyyzz.LIB`

Symbol in CODESYS:



Beschreibung

2374

MEMORY_RETAIN_PARAM legt das remanente Verhalten der Daten für verschiedene Ereignisse fest. In CODESYS als VAR_RETAIN deklarierte Variablen haben von vornherein ein remanentes Verhalten.

Remanente Daten behalten (wie die als VAR_RETAIN deklarierte Variablen) ihren Wert nach einem unkontrolliertem Beenden wie auch nach normalem Aus- und Einschalten der Steuerung. Bei erneutem Start arbeitet das Programm mit den gespeicherten Werten weiter.

Für (mit MODE) wählbare Gruppen von Ereignissen legt dieser FB fest, wie viele (LEN) Datenbytes (ab Merkerbyte %MB0) Retain-Verhalten haben sollen, auch wenn sie nicht ausdrücklich als VAR_RETAIN deklariert wurden.

Ereignis	MODE = 0	MODE = 1	MODE = 2	MODE = 3
Power OFF ⇨ ON	Daten werden neu initialisiert	Daten sind remanent	Daten sind remanent	Daten sind remanent
Reset warm	Daten werden neu initialisiert	Daten sind remanent	Daten sind remanent	Daten sind remanent
Reset kalt	Daten werden neu initialisiert	Daten werden neu initialisiert	Daten sind remanent	Daten sind remanent
Reset Ursprung	Daten werden neu initialisiert	Daten werden neu initialisiert	Daten sind remanent	Daten sind remanent
Anwendungsprogramm laden	Daten werden neu initialisiert	Daten werden neu initialisiert	Daten sind remanent	Daten sind remanent
Laufzeitsystem laden	Daten werden neu initialisiert	Daten werden neu initialisiert	Daten werden neu initialisiert	Daten sind remanent

Bei MODE = 0 habe nur solche Daten Retain-Verhalten wie bei MODE=1, die ausdrücklich als VAR_RETAIN deklariert wurden.

Parameter der Eingänge

2375

Parameter	Datentyp	Beschreibung
ENABLE	BOOL	TRUE: Baustein ausführen FALSE: Baustein wird nicht ausgeführt > Baustein-Eingänge sind nicht aktiv > Baustein-Ausgänge sind nicht spezifiziert
LEN	WORD	Anzahl der Datenbytes ab Merkeradresse %MB0, die remanentes Verhalten haben sollen zulässig → FB-Beschreibung in der Gerätebibliothek
MODE	BYTE	Ereignisse, bei denen diese Variablen Retain-Verhalten haben sollen (0...3; → Tabelle oben)

Manuelle Datensicherung

Inhalt

13792

Neben der Möglichkeit, die Daten automatisch zu sichern, können über FB-Aufrufe Anwenderdaten manuell in integrierte Speicher gesichert und von dort wieder gelesen werden.

► Sicherheitsrelevante Daten immer mit geeigneten Methoden sichern (z.B. → ***CHECK_DATA*** (→ Seite 375))!

Der Programmierer kann sich anhand der Speicheraufteilung (→ Kapitel ***Verfügbarer Speicher*** (→ Seite 129)) darüber informieren, welcher Speicherbereich frei zur Verfügung steht.

FLASHREAD

561

Baustein-Typ = Funktionsbaustein (FB)

Baustein ist enthalten in Bibliothek `ifm_CR7132_Vxxyyzz.LIB`

Symbol in CODESYS:

FLASHREAD
ENABLE
SRC
LEN
DST

Beschreibung

564

FLASHREAD ermöglicht das Lesen unterschiedlicher Datentypen direkt aus dem Flash-Speicher in den RAM.

> Der FB liest den Inhalt ab der Adresse von SRC aus dem Flash-Speicher. Dabei werden genau so viele Bytes übertragen, wie diese unter LEN angegeben sind.

> Das Lesen erfolgt komplett in dem Zyklus, in dem der FB aufgerufen wird.

► Darauf achten, dass der Zielspeicherbereich im RAM groß genug ist.

► Für die Zieladresse DST gilt:
(!) Die Adresse mit dem Operator ADR ermitteln und dem FB übergeben!

Parameter der Eingänge

2318

Parameter	Datentyp	Beschreibung
ENABLE	BOOL	TRUE: Baustein ausführen FALSE: Baustein wird nicht ausgeführt > Baustein-Eingänge sind nicht aktiv > Baustein-Ausgänge sind nicht spezifiziert
SRC	DWORD	Relative Anfangsadresse im Speicher zulässig = 0...$65\,535_{10}$ = 0...0000 $FFFF_{16}$ (!) Falls Startadresse außerhalb des zulässigen Bereichs: > kein Datentransfer
LEN	DWORD	Anzahl der Datenbytes (max. 65 536 = 0x0001 0000) (!) Würde durch die angegebene Anzahl an Bytes der Flash-Speicherbereich überschritten werden, werden die Daten nur bis zum Ende des Flash-Speicherbereichs übertragen.
DST	DWORD	Startadresse im Zielspeicher (!) Die Adresse mit dem Operator ADR ermitteln und dem FB übergeben!

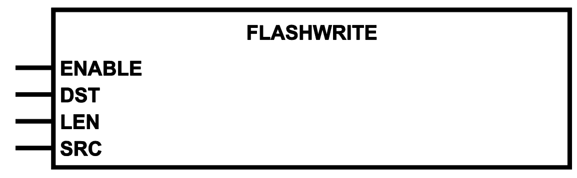
FLASHWRITE

555

Baustein-Typ = Funktionsbaustein (FB)

Baustein ist enthalten in Bibliothek `ifm_CR7132_Vxxyyzz.LIB`

Symbol in CODESYS:



Beschreibung

12892

► Für den Einsatz des FBs den TEST-Eingang aktivieren! Ansonsten wird der Aufruf ignoriert.

> Test-Eingang ist aktiv:
• Programmiermodus ist freigeben
• Software-Download ist möglich
• die sicheren Ausgänge sind deaktiviert
• es werden keine CANsafety-Nachrichten versendet
• Zustand des Anwendungsprogramms ist abfragbar
• kein Schutz der gespeicherten Software möglich

558

⚠ WARNUNG

Gefahr durch unkontrollierten Prozessablauf!

Der Zustand der Ein-/Ausgänge wird während der Ausführung von FLASHWRITE "eingefroren".

► Diesen Funktionsbaustein nicht bei laufender Maschine ausführen!

FLASHWRITE ermöglicht das Schreiben unterschiedlicher Datentypen direkt in den Flash-Speicher.

Mit diesem FB sollen während der Inbetriebnahme große Datenmengen gesichert werden, auf die im Prozess nur lesend zugegriffen wird.

► Vor erneutem Beschreiben des Flash-Speichers den kompletten Flash-Speicher löschen!
Dies geschieht mit dem Beschreiben der Adresse "0" mit beliebigem Inhalt.
Den Flash-Speicherbereich nicht öfter als 100mal löschen, da ansonsten die Datenkonsistenz in anderen Flash-Speicherbereichen nicht mehr gewährleistet werden kann.

► In jedem SPS-Zyklus darf FLASHWRITE nur einmalig gestartet werden!

► Für die Zieladresse DST gilt:
Die Adresse mit dem Operator ADR ermitteln und dem FB übergeben!

> Der FB schreibt den Inhalt der Adresse SRC in den Flash-Speicher. Dabei werden genau so viele Bytes übertragen, wie diese unter LEN angegeben sind.

Falls Startadresse SRC außerhalb des zulässigen Bereichs: kein Datentransfer!

Parameter der Eingänge

2603

Parameter	Datentyp	Beschreibung
ENABLE	BOOL	TRUE: Baustein ausführen FALSE: Baustein wird nicht ausgeführt > Baustein-Eingänge sind nicht aktiv > Baustein-Ausgänge sind nicht spezifiziert
DST	DWORD	Relative Anfangsadresse im Speicher zulässig = 0...65 534_{10} = 0...0000 $FFFE_{16}$ Die Adresse mit dem Operator ADR ermitteln und dem FB übergeben!
LEN	DWORD	Anzahl der Datenbytes (max. 65 536 = 0x0001 0000) Würde durch die angegebene Anzahl an Bytes der Flash-Speicherbereich überschritten werden, werden die Daten nur bis zum Ende des Flash-Speicherbereichs übertragen.
SRC	DWORD	Quell-Adresse

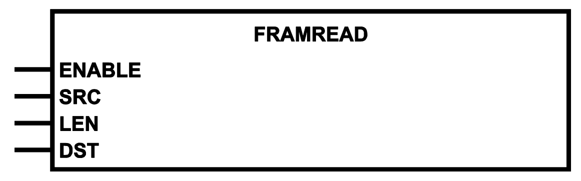
FRAMREAD

549

Baustein-Typ = Funktionsbaustein (FB)

Baustein ist enthalten in Bibliothek `ifm_CR7132_Vxxyyzz.LIB`

Symbol in CODESYS:



Beschreibung

552

FRAMREAD ermöglicht das schnelle Lesen unterschiedlicher Datentypen direkt aus dem Anwender-Retain-Speicher (FRAM[1]).

Der FB liest den Inhalt ab der Adresse von SRC aus dem FRAM-Speicher. Dabei werden genau so viele Bytes übertragen, wie diese unter LEN angegeben sind.

► Für die Zieladresse DST gilt:
  Die Adresse mit dem Operator ADR ermitteln und dem FB übergeben!

Der FRAM-Speicher kann in mehreren unabhängigen Teilsegmenten ausgelesen werden. Die Überwachung der Speichersegmente muss im Anwendungsprogramm erfolgen.

[1]) FRAM steht hier allgemein für alle Arten von nichtflüchtigen, schnellen Speichern.

Parameter der Eingänge

2606

Parameter	Datentyp	Beschreibung
ENABLE	BOOL	TRUE: Baustein ausführen FALSE: Baustein wird nicht ausgeführt > Baustein-Eingänge sind nicht aktiv > Baustein-Ausgänge sind nicht spezifiziert
SRC	DWORD	Relative Anfangsadresse im Speicher zulässig = 0... 16 383_{10} = 0...0000 $3FFF_{16}$
LEN	DWORD	Anzahl der Datenbytes zulässig = 0...16 384_{10} = 0...0000 4000_{16}
DST	DWORD	Startadresse im Zielspeicher Die Adresse mit dem Operator ADR ermitteln und dem FB übergeben!

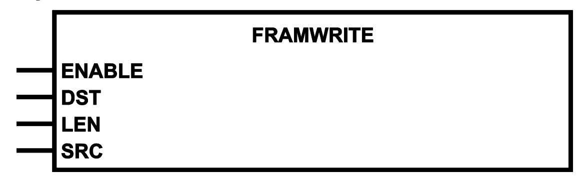
FRAMWRITE

543

Baustein-Typ = Funktionsbaustein (FB)

Baustein ist enthalten in Bibliothek `ifm_CR7132_Vxxyyzz.LIB`

Symbol in CODESYS:



Beschreibung

546

FRAMWRITE ermöglicht das schnelle Schreiben unterschiedlicher Datentypen direkt in den Anwender-Retain-Speicher (FRAM[1]).

Der FB schreibt den Inhalt ab der Adresse SRC in den spannungsausfallsicheren FRAM-Speicher. Dabei werden genau so viele Bytes übertragen, wie diese über LEN angegeben sind.

► Für die Quelladresse SRC gilt:
(!) Die Adresse mit dem Operator ADR ermitteln und dem FB übergeben!

Der FRAM-Speicher kann in mehreren unabhängigen Teilsegmenten beschrieben werden. Die Überwachung der Speichersegmente muss im Anwendungsprogramm erfolgen.

[1]) FRAM steht hier allgemein für alle Arten von nichtflüchtigen, schnellen Speichern.

Parameter der Eingänge

2605

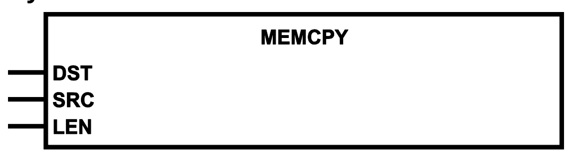
Parameter	Datentyp	Beschreibung
ENABLE	BOOL	TRUE: Baustein ausführen FALSE: Baustein wird nicht ausgeführt > Baustein-Eingänge sind nicht aktiv > Baustein-Ausgänge sind nicht spezifiziert
DST	DWORD	Relative Zieladresse im Speicher zulässig = 0...16 383$_{10}$ = 0...0000 3FFF$_{16}$
LEN	DWORD	Anzahl der Datenbytes zulässig = 0...16 384$_{10}$ = 0...0000 4000$_{16}$
SRC	DWORD	Startadresse im Quellspeicher (!) Die Adresse mit dem Operator ADR ermitteln und dem FB übergeben!

MEMCPY

409

= Memory Copy
Baustein-Typ = Funktionsbaustein (FB)
Baustein ist enthalten in Bibliothek `ifm_CR7132_Vxxyyzz.LIB`

Symbol in CODESYS:



Beschreibung

412

MEMCPY ermöglicht das Schreiben und Lesen unterschiedlicher Datentypen direkt in den Speicher.
Der FB schreibt den Inhalt ab der Adresse von SRC an die Adresse DST.

► Für die Adressen SRC und DST gilt:
  ! Die Adresse mit dem Operator ADR ermitteln und dem FB übergeben!

> Dabei werden genau so viele Bytes übertragen, wie diese unter LEN angegeben wurden. Dadurch ist es auch möglich, genau ein Byte einer Word-Variablen zu übertragen.

Parameter der Eingänge

413

Parameter	Datentyp	Beschreibung
DST	DWORD	Startadresse im Zielspeicher ! Die Adresse mit dem Operator ADR ermitteln und dem FB übergeben!
SRC	DWORD	Startadresse im Quellspeicher ! Die Adresse mit dem Operator ADR ermitteln und dem FB übergeben!
LEN	WORD	Anzahl (≥ 1) der zu übertragenden Daten-Bytes

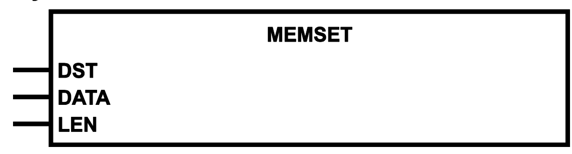
MEMSET

2348

Baustein-Typ = Funktionsbaustein (FB)

Baustein ist enthalten in Bibliothek `ifm_CR7132_Vxxyyzz.LIB`

Symbol in CODESYS:



Beschreibung

2350

MEMSET ermöglicht das Beschreiben eines bestimmten Datenbereiches.

Der FB beschreibt den Speicher ab der Adresse DST mit der Anzahl von LEN Bytes mit dem Inhalt von DATA.

► Für die Ziel-Adresse DST gilt:
  ! Die Adresse mit dem Operator ADR ermitteln und dem FB übergeben!

Parameter der Eingänge

2351

Parameter	Datentyp	Beschreibung
DST	DWORD	Startadresse im Zielspeicher ! Die Adresse mit dem Operator ADR ermitteln und dem FB übergeben!
DATA	BYTE	zu schreibender Wert
LEN	WORD	Anzahl der mit DATA zu beschreibenden Datenbytes

6.2.21 Bausteine: Datenzugriff und Datenprüfung

Inhalt

1598

Die Bausteine in diesem Kapitel steuern den Datenzugriff und ermöglichen ein Prüfen der Daten.

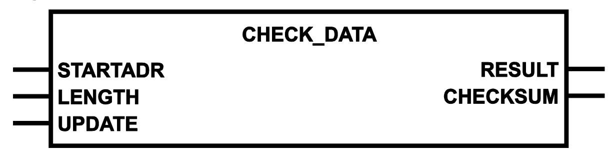
CHECK_DATA

603

Baustein-Typ = Funktionsbaustein (FB)

Baustein ist enthalten in Bibliothek `ifm_CR7132_Vxxyyzz.LIB`

Symbol in CODESYS:



Beschreibung

606

CHECK_DATA erzeugt über einen konfigurierbaren Speicherbereich eine Prüfsumme (CRC) und prüft die Daten des Speicherbereichs auf ungewollte Veränderung.

► Für jeden zu überwachenden Speicherbereich eine eigene Instanz des FB erzeugen.

► (!) Die Adresse mit dem Operator ADR ermitteln und dem FB übergeben!

► Zusätzlich die Anzahl der Datenbytes LENGTH (Länge ab der STARTADR) angeben.

Ungewollte Änderung: Fehler!
Wenn Eingang UPDATE = FALSE und Daten im Speicher sich ungewollt verändern, wird RESULT = FALSE. Das Ergebnis kann dann für weitere Aktionen (z.B. Abschalten der Ausgänge) genutzt werden.

Gewollte Änderung:
Nur wenn der Eingang UPDATE auf TRUE gesetzt ist, sind Datenänderungen im Speicher (z.B. vom Anwendungsprogramm oder **ecomat*mobile***-Gerät) zulässig. Der Wert der Prüfsumme wird dann neu berechnet. Der Ausgang RESULT ist wieder permanent TRUE.

Parameter der Eingänge

2612

Parameter	Datentyp	Beschreibung
STARTADR	DWORD	Startadresse des überwachten Datenspeichers (WORD-Adresse ab %MW0) (!) Die Adresse mit dem Operator ADR ermitteln und dem FB übergeben!
LENGTH	DWORD	Länge des überwachten Datenspeichers in [Byte]
UPDATE	BOOL	TRUE: Daten wurden geändert > FB berechnet eine neue Prüfsumme FALSE: Daten wurden nicht geändert > FB prüft den Speicherbereich

Parameter der Ausgänge

2613

Parameter	Datentyp	Beschreibung
RESULT	BOOL	TRUE: CRC-Prüfsumme in Ordnung: Daten sind gewollt verändert oder nicht verändert FALSE: CRC-Prüfsumme fehlerhaft: Daten wurden ungewollt verändert
CHECKSUM	DWORD	aktuelle CRC-Prüfsumme

Beispiel: CHECK_DATA

4168

Im folgenden Beispiel ermittelt das Programm die Prüfsumme und legt sie über den Pointer pt im RAM ab:

```
PROGRAM PLC_PRG
VAR
    m1 : BOOL := TRUE;
    cd1 : CHECK_DATA;
    ok : BOOL;
    pt : POINTER TO WORD;
END_VAR
```

```
0001
16#82DC00 ─┤ADD├─┤SUB├── pt
16#400    ─┘   2 ─┘

0002
                                  cd1
16#400 ─┤SUB├── 16#82DC00 ─┤CHECK_DATA        ├
     2 ─┘                  ─┤STARTADR   RESULT├── ok
                           ─┤LENGTH   CHECKSUM├── pt
                       m1 ─┤UPDATE             │

0003
FALSE ── m1
```

Das hier gezeigte Verfahren ist für den Flash-Speicher nicht geeignet.

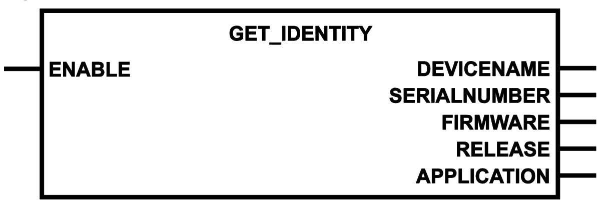
GET_IDENTITY

14505

Baustein-Typ = Funktionsbaustein (FB)

Baustein ist enthalten in Bibliothek `ifm_CR7132_Vxxyyzz.LIB`

Symbol in CODESYS:



Beschreibung

14507

GET_IDENTITY liest die im Gerät gespeicherten spezifischen Kennungen:
- Hardware-Name und Hardware-Version des Geräts
- Seriennummer des Geräts
- Name des Laufzeitsystems im Gerät
- Version und Ausgabe des Laufzeitsystems im Gerät
- Name der Anwendung (wurde zuvor mit ***SET_IDENTITY*** (→ Seite 379) gespeichert)

Parameter der Eingänge

2609

Parameter	Datentyp	Beschreibung
ENABLE	BOOL	TRUE: Baustein ausführen FALSE: Baustein wird nicht ausgeführt > Baustein-Eingänge sind nicht aktiv > Baustein-Ausgänge sind nicht spezifiziert

Parameter der Ausgänge

14508

Parameter	Datentyp	Beschreibung
DEVICENAME	STRING(31)	Hardware-Name und Hardware-Version des Geräts als Zeichenkette von max. 31 Zeichen z.B.: "CR0403 01.00.00"
SERIALNUMBER	STRING(31)	Seriennummer des Geräts als Zeichenkette von max. 31 Zeichen z.B.: "12345678"
FIRMWARE	STRING(31)	Name des Laufzeitsystems im Gerät als Zeichenkette von max. 31 Zeichen z.B.: "CR0403"
RELEASE	STRING(31)	Version und Ausgabe des Laufzeitsystems im Gerät als Zeichenkette von max. 31 Zeichen z.B.: "V01.00.00 120215"
APPLICATION	STRING(79)	Name der Anwendung als String von max. 79 Zeichen z.B.: "Crane1704"

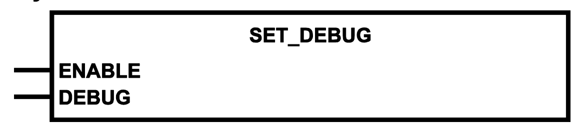
SET_DEBUG

290

Baustein-Typ = Funktionsbaustein (FB)

Baustein ist enthalten in Bibliothek `ifm_CR7132_Vxxyyzz.LIB`

Symbol in CODESYS:



Beschreibung

13045

SET_DEBUG organisiert (abhängig vom TEST-Eingang) den DEBUG-Modus oder den Monitoring-Modus (→ Kapitel ***TEST-Betrieb*** (→ Seite 43)):

Controller-Eingang TEST	FB SET_DEBUG Eingang DEBUG	Variablenwerte	sichere Ausgänge
nicht mit VBB verbunden	TRUE	MONITORING-Modus: nur lesen möglich	aktiv
mit VBB verbunden	TRUE oder FALSE	DEBUG-Modus: lesen und ändern möglich	deaktiviert

Wird (bei offenem TEST-Eingang!) der FB-Eingang DEBUG=TRUE gesetzt, kann z.B. das Programmiersystem oder der Downloader mit dem Gerät kommunizieren und Geräteinformationen und Variablenwerte der Anwendung auslesen.

Ein Software-Download ist im MONITORING-Betrieb nicht möglich, da der Test-Eingang nicht mit Versorgungsspannung verbunden ist.

HINWEIS

Der Debug-Modus kann bestehen bleiben, auch wenn das Anwendungsprogramm ohne erneutes Verwenden von SET_DEBUG aktualisiert wurde.

> Ein fortgesetzter Lesezugriff kann möglich sein.
> Ein fortgesetzter Schreibzugriff ist nicht mehr möglich.

► Nach Aktualisieren des Anwendungsprogramms ein Power-On-Reset durchführen! Somit wird der Debug-Modus zuverlässig unterbrochen.

Parameter der Eingänge

13046

Parameter	Datentyp	Beschreibung
ENABLE	BOOL	TRUE: Baustein ausführen FALSE: Baustein wird nicht ausgeführt > Baustein-Eingänge sind nicht aktiv > Baustein-Ausgänge sind nicht spezifiziert
DEBUG	BOOL	TRUE: Monitoring über die Schnittstellen möglich (nur lesend) FALSE: Monitoring über die Schnittstellen nicht möglich

SET_IDENTITY

11927

Baustein-Typ = Funktionsbaustein (FB)

Baustein ist enthalten in Bibliothek `ifm_CR7132_Vxxyyzz.LIB`

Symbol in CODESYS:



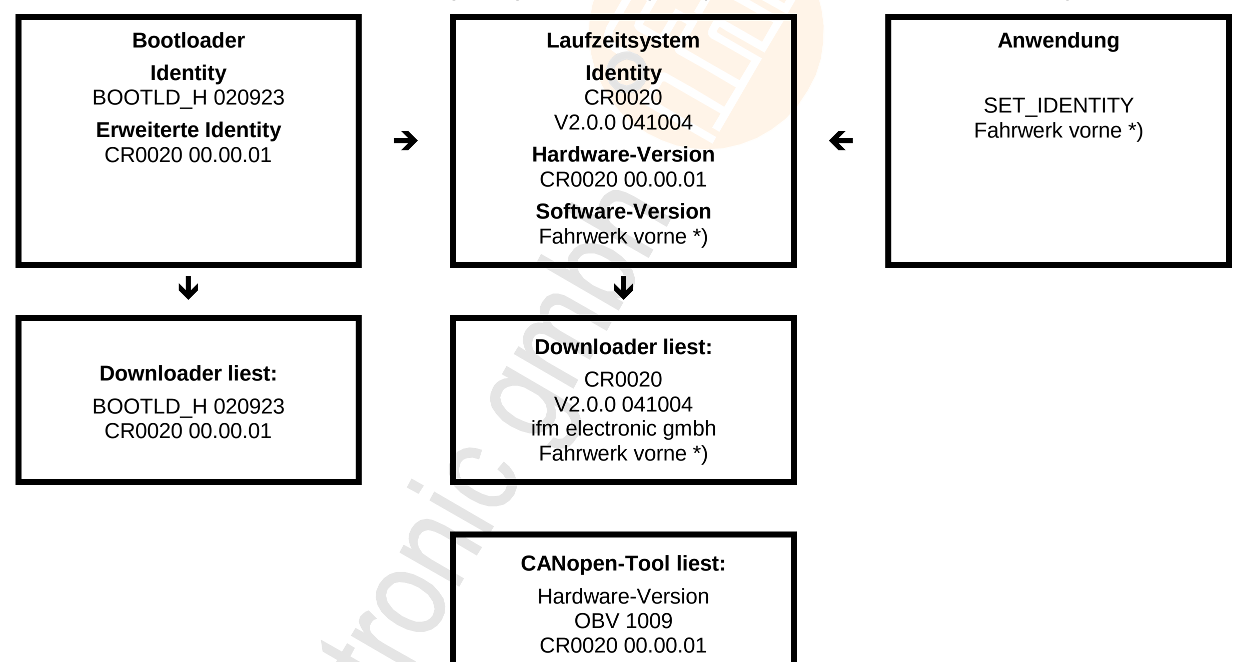
Beschreibung

287

SET_IDENTITY setzt eine anwendungsspezifische Programmkennung.

Mit dem FB kann durch das Anwendungsprogramm eine Programmkennung erzeugt werden. Diese Kennung kann zur Identifizierung des geladenen Programms über das Software-Tool DOWNLOADER.EXE als Software-Version ausgelesen werden.

Die nachfolgende Grafik zeigt die Zusammenhänge der unterschiedlichen Kennungen, wie sie mit den unterschiedlichen Software-Tools angezeigt werden. (Beispiel: ClassicController CR0020):



*) 'Fahrwerk vorne' steht hier stellvertretend für einen kundenspezifischen Text.

Parameter der Eingänge

11928

Parameter	Datentyp	Beschreibung
ENABLE	BOOL	TRUE: Baustein ausführen FALSE: Baustein wird nicht ausgeführt > Baustein-Eingänge sind nicht aktiv > Baustein-Ausgänge sind nicht spezifiziert
ID	STRING(79)	beliebiger Text mit einer maximalen Länge von 79 Zeichen

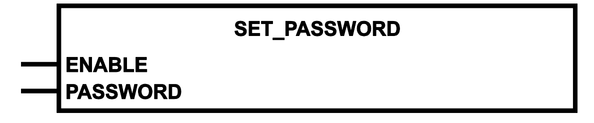
SET_PASSWORD

266

Baustein-Typ = Funktionsbaustein (FB)

Baustein ist enthalten in Bibliothek `ifm_CR7132_Vxxyyzz.LIB`

Symbol in CODESYS:



Beschreibung

269

SET_PASSWORD setzt Benutzerkennung für Programm- und Speicher-Upload mit dem DOWNLOADER.

Ist die Benutzerkennung aktiv, kann durch das Software-Tool DOWNLOADER das Anwendungsprogramm oder der Datenspeicher nur ausgelesen werden, wenn das richtige Password eingegeben wurde.

Wird an den Eingang PASSWORD ein Leer-String (Default-Zustand) übergeben, ist ein Upload des Anwendungsprogramms oder des Datenspeichers jederzeit möglich.

> ! Beim Laden eines neuen Anwendungsprogramms wird die Kennung wieder zurückgesetzt.

Parameter der Eingänge

13040

Parameter	Datentyp	Beschreibung
ENABLE	BOOL	TRUE (nur 1 Zyklus lang): Baustein ausführen FALSE: im weiteren Programmablauf
PASSWORD	STRING(16)	Benutzerkennung Wenn PASSWORD = "", dann ist Zugriff ohne Passworteingabe möglich.

6.2.22 Bausteine: Fehlermeldungen verwalten

Inhalt

13797

Hier zeigen wir Ihnen Funktionen, mit denen Sie Folgendes erreichen:
- anwendungsspezifische Fehler-Codes generieren
- Fehler-Codes auflisten oder löschen
- Ausgänge und CANsafety-Nachrichtenkanäle von Fehlerreaktion ausnehmen

ERROR_REPORT

12357

Baustein-Typ = Funktionsbaustein (FB)

Baustein ist enthalten in Bibliothek `ifm_CR7132_Vxxyyzz.LIB`

Symbol in CODESYS:



Beschreibung

12364

Mit ERROR_REPORT meldet das Anwendungsprogramm dem System einen anwendungsspezifischen Fehler.

► Das Ergebnis der Fehlerbedingung auf den Eingang ENABLE programmieren.
Bei ENABLE=TRUE trägt der FB den Fehler-Code in die Fehlerliste ein.
 • aktuelle Fehlerliste ansehen mit ***SHOW_ERROR_LIST*** (→ Seite 389)
 • einzelnen Eintrag aus der Fehlerliste löschen mit ***ERROR_RESET*** (→ Seite 384)

► Den zugehörigen Fehler-Code auf den Eingang ERRORCODE programmieren:
Muster: yy xx 00 00 → Kapitel ***Fehler-Codes*** (→ Seite 392)
xx = anwendungsspezifischer Fehler-Code
yy = Fehlerklasse
Dazu hilfreich ist der FB ***PACK_ERRORCODE*** (→ Seite 388)

> Der FB-Ausgang ERROR meldet, ob dieser FB-Aufruf richtig parametriert wurde.
(!) Der FB prüft die Fehler-Codes nicht darauf, ob sie sinnvoll sind.

Parameter der Eingänge

12363

Parameter	Datentyp	Beschreibung
ENABLE	BOOL	TRUE: Baustein ausführen ein Fehler ist aktiv den dazu definierten ERRORCODE melden FALSE: Baustein wird nicht ausgeführt der Fehler ist nicht (mehr) aktiv
ERRORCODE	DWORD	Fehler-Code, bei dessen Auftreten das konfigurierte Verhalten angewendet werden soll. → Kapitel ***Fehler-Codes*** (→ Seite 392) (!) Der FB prüft die Fehler-Codes nicht darauf, ob sie sinnvoll sind.

Parameter der Ausgänge

12365

Parameter	Datentyp	Beschreibung
ERROR	DWORD	Fehler-Code aus diesem FB-Aufruf → ***Fehler-Codes*** (→ Seite 392) (mögliche Meldungen → folgende Tabelle)

Mögliche Ergebnisse für ERROR (n=beliebiger Wert):

Der 32-Bit-Fehler-Code besteht aus vier 8-Bit-Werten (DWORD).

4. Byte	3. Byte	2. Byte	1. Byte
Fehlerklasse	anwendungsspezifischer Fehler-Code	Fehlerquelle	Fehlerursache

Wert [hex]	Beschreibung
00 00 00 00	kein Fehler
02 00 00 F8	falscher Parameter ⇨ schwerer Fehler

ERROR_RESET

12376

Baustein-Typ = Funktionsbaustein (FB)

Baustein ist enthalten in Bibliothek `ifm_CR7132_Vxxyyzz.LIB`

Symbol in CODESYS:



Beschreibung

12378

Mit ERROR_RESET kann das Anwendungsprogramm anstehende Fehlermeldungen zurücksetzen:
- eine einzelne Fehlermeldung
- eine Gruppe gleichartiger Fehlermeldungen (gleiche Quelle oder gleiche Ursache)
- alle Fehler

Solange ein Fehler aktiv ist, kann dessen Fehlermeldung nicht zurückgesetzt werden. Diese Fehlermeldung bleibt trotz Rücksetzkommando weiterhin in der Fehlerliste und wirksam.

► Zwischen zwei Rücksetz-Aktionen des FBs:
ENABLE=FALSE setzen (mindestens einen SPS-Zyklus lang)!

Parameter der Eingänge

12379

Parameter	Datentyp	Beschreibung
ENABLE	BOOL	TRUE: Baustein ausführen FALSE: Baustein wird nicht ausgeführt > Baustein-Eingänge sind nicht aktiv > Baustein-Ausgänge sind nicht spezifiziert
ERRORCODE	DWORD	Fehler-Code, bei dessen Auftreten das konfigurierte Verhalten angewendet werden soll. → Kapitel ***Fehler-Codes*** (→ Seite 392) (!) Der FB prüft die Fehler-Codes nicht darauf, ob sie sinnvoll sind.

Parameter der Ausgänge

12380

Parameter	Datentyp	Beschreibung
ERROR	DWORD	Fehler-Code aus diesem FB-Aufruf → ***Fehler-Codes*** (→ Seite 392) (mögliche Meldungen → folgende Tabelle)

Mögliche Ergebnisse für ERROR (n=beliebiger Wert):

Der 32-Bit-Fehler-Code besteht aus vier 8-Bit-Werten (DWORD).

4. Byte	3. Byte	2. Byte	1. Byte
Fehlerklasse	anwendungsspezifischer Fehler-Code	Fehlerquelle	Fehlerursache

Wert [hex]	Beschreibung
00 00 00 00	kein Fehler
02 00 00 F8	falscher Parameter ⇨ schwerer Fehler

Beispiel: ERROR_RESET

13054

Sollen alle Fehler "Überlast" mit der Fehlerklasse "Allgemeiner Fehler" zurückgesetzt werden, dann muss ERRORCODE=0x01000004 angegeben werden:

4. Byte	3. Byte	2. Byte	1. Byte
Fehlerklasse	anwendungsspezifischer Fehler-Code	Fehlerquelle	Fehlerursache
0x01	0x00	0x00	0x04
allgemeiner Fehler	kein anwendungsspezifischer Fehler	alle Fehlerquellen	Überlast

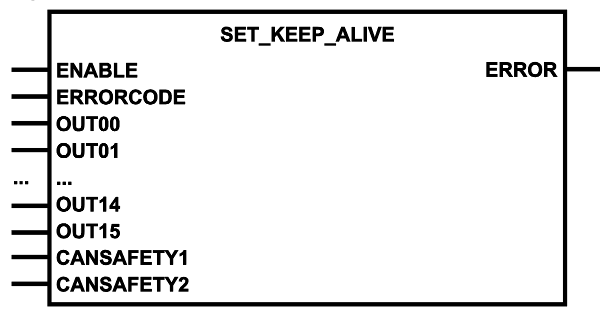
SET_KEEP_ALIVE

12346

Baustein-Typ = Funktionsbaustein (FB)

Baustein ist enthalten in Bibliothek `ifm_CR7132_Vxxyyzz.LIB`

Symbol in CODESYS:



Beschreibung

12348

Ausführliche Beschreibung → Kapitel ***Keep-Alive-Funktionalität*** (→ Seite 74)

Mit SET_KEEP_ALIVE konfigurieren, welche Ausgangskanäle und welche CANsafety-Schnittstellen beim Auftreten eines bestimmten schweren Fehlers weiterbetrieben werden sollen, da sie von dem mit ERRORCODE gemeldeten Fehler unabhängig sind.

> Wird bei der Konfiguration der Fehler-Code eines Systemfehlers übergeben, setzt der FB dessen Fehlerklasse automatisch auf den Wert 0x00 (keine Fehlerklasse).

> Wird bei der Konfiguration der Fehler-Code eines Anwendungsfehlers übergeben, setzt der FB dessen Fehlerklasse automatisch auf den Wert 0x02 (schwerer Fehler).

Beides ist am Eingang ERRORCODE des FBs sichtbar.

Parameter der Eingänge

12349

Parameter	Datentyp	Beschreibung
ENABLE	BOOL	TRUE: Baustein ausführen FALSE: Baustein wird nicht ausgeführt > Baustein-Eingänge sind nicht aktiv > Baustein-Ausgänge sind nicht spezifiziert
ERRORCODE	DWORD	Fehler-Code, bei dessen Auftreten das konfigurierte Verhalten angewendet werden soll. → Kapitel ***Fehler-Codes*** (→ Seite 392) (!) Der FB prüft die Fehler-Codes nicht darauf, ob sie sinnvoll sind.
OUTxx	BOOL	Ausgang xx (xx=00...15), wenn als sicherer Ausgang konfiguriert, soll beim Auftreten des Fehlers ERRORCODE weiterbetrieben werden und nicht in den sicheren Zustand gehen. TRUE: Ausgang wird weiterbetrieben FALSE: Ausgang geht in den sicheren Zustand
CANSAFETYx	BOOL	CANsafety-Kommunikation mit anderen Geräten über den CANsafety-Kanal x (x=1...2) soll beim Auftreten des Fehlers ERRORCODE weiterbetrieben werden und nicht in den sicheren Zustand gehen. TRUE: CANsafety-Kanal wird weiterbetrieben FALSE: CANsafety-Kanal geht in den sicheren Zustand

Parameter der Ausgänge

12350

Parameter	Datentyp	Beschreibung
ERROR	DWORD	Fehler-Code aus diesem FB-Aufruf → ***Fehler-Codes*** (→ Seite 392) (mögliche Meldungen → folgende Tabelle)

Mögliche Ergebnisse für ERROR (n=beliebiger Wert):

Der 32-Bit-Fehler-Code besteht aus vier 8-Bit-Werten (DWORD).

4. Byte	3. Byte	2. Byte	1. Byte
Fehlerklasse	anwendungsspezifischer Fehler-Code	Fehlerquelle	Fehlerursache

Wert [hex]	Beschreibung
00 00 00 00	kein Fehler

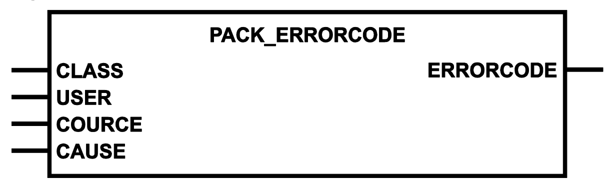
PACK_ERRORCODE

12382

Baustein-Typ = Funktionsbaustein (FB)

Baustein ist enthalten in Bibliothek `ifm_CR7132_Vxxyyzz.LIB`

Symbol in CODESYS:



Beschreibung

12384

PACK_ERRORCODE hilft beim Zusammenbauen eines ERRORCODE aus den Bestandteilen:

- Fehlerklasse
- anwendungsspezifischer Fehler
- Fehlerquelle
- Fehlerursache

(Struktur → Kapitel ***Fehler-Codes*** (→ Seite 392)).

❗ Der FB prüft die Fehler-Codes nicht darauf, ob sie sinnvoll sind.

Parameter der Eingänge

12385

Parameter	Datentyp	Beschreibung
CLASS	BYTE	Code für Fehlerklasse
USER	BYTE	Code für anwendungsspezifischer Fehler
SOURCE	BYTE	Code für Fehlerquelle
CAUSE	BYTE	Code für Fehlerursache

Parameter der Ausgänge

12390

Parameter	Datentyp	Beschreibung
ERRORCODE	DWORD	Fehler-Code → Kapitel ***Fehler-Codes*** (→ Seite 392) ❗ Der FB prüft die Fehler-Codes nicht darauf, ob sie sinnvoll sind.

SHOW_ERROR_LIST

12360

Baustein-Typ = Funktionsbaustein (FB)

Baustein ist enthalten in Bibliothek `ifm_CR7132_Vxxyyzz.LIB`

Symbol in CODESYS:



Beschreibung

12367

Der FB SHOW_ERROR_LIST dient zum Auslesen der aktuell vorliegenden Fehler-Codes.

Mit ENABLE=TRUE erstellt der FB eine Liste von bis zu 64 derzeit aktuellen Fehler-Codes.
Bei ENABLE=FALSE bleibt die zuletzt erstellte Liste unverändert erhalten.

Ist die Liste voll, werden keine weiteren Fehler-Codes mehr aufgenommen.

Parameter der Eingänge

12368

Parameter	Datentyp	Beschreibung
ENABLE	BOOL	TRUE: Baustein ausführen FALSE: Baustein wird nicht ausgeführt > Baustein-Eingänge sind nicht aktiv > Baustein-Ausgänge sind nicht spezifiziert

Parameter der Ausgänge

12369

Parameter	Datentyp	Beschreibung
ERRORS	ARRAY [0..63] OF DWORD	Liste mit den aktuell vorliegenden Fehler-Codes → Kapitel ***Fehler-Codes*** (→ Seite 392)

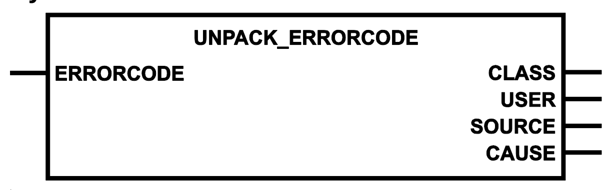
UNPACK_ERRORCODE

13650

Baustein-Typ = Funktionsbaustein (FB)

Baustein ist enthalten in Bibliothek `ifm_CR7132_Vxxyyzz.LIB`

Symbol in CODESYS:



Beschreibung

13653

UNPACK_ERRORCODE trennt einen ERRORCODE in seine Bestandteile:
- Fehlerklasse
- anwendungsspezifischer Fehler
- Fehlerquelle
- Fehlerursache

(Struktur → Kapitel ***Fehler-Codes*** (→ Seite 392)).

! Der FB prüft die Fehler-Codes nicht darauf, ob sie sinnvoll sind.

Parameter der Eingänge

13654

Parameter	Datentyp	Beschreibung
ERRORCODE	DWORD	Fehler-Code → Kapitel ***Fehler-Codes*** (→ Seite 392) ! Der FB prüft die Fehler-Codes nicht darauf, ob sie sinnvoll sind.

Parameter der Ausgänge

13675

Parameter	Datentyp	Beschreibung
CLASS	BYTE	Code für Fehlerklasse
USER	BYTE	Code für anwendungsspezifischer Fehler
SOURCE	BYTE	Code für Fehlerquelle
CAUSE	BYTE	Code für Fehlerursache

7 Fehler-Codes und Diagnoseinformationen

Inhalt

14024

7.1 Übersicht

14377

Werden bei der Systemüberwachung Fehler erkannt, reagiert die Steuerung darauf. Je nach Schwere unterscheidet sich das Verhalten der Steuerung.

Wir unterscheiden:
- allgemeine Fehler
- schwere Fehler
- fatale Fehler

⚠ WARNUNG

Gefahr durch unbeabsichtigtes und gefährliches Anlaufen von Maschinen- oder Anlagenteilen!

► Der Programmierer muss bei der Programmerstellung verhindern, dass nach Auftreten eines Fehlers (z.B. NOT-HALT) und der anschließenden Fehlerbeseitigung unbeabsichtigt Maschinen- oder Anlagenteile gefährlich anlaufen können!
⇨ Wiederanlaufsperre realisieren!

► Dazu im Fehlerfall die in Frage kommenden Ausgänge im Programm logisch abschalten!

Beim Zurücksetzen eines Fehler-Codes mittels ***ERROR_RESET*** (→ Seite 384) wird auch der dazugehörende Fehlermerker zurückgesetzt.

Zusätzlich besteht auch die Möglichkeit, mittels ***ERROR_REPORT*** (→ Seite 382) "frei definierte Fehler" im Anwendungsprogramm zu setzen.

→ auch Kapitel ***Systemmerker*** (→ Seite 420)

7.2 Fehler-Codes

Inhalt

12334

Übersicht der Fehler-Codes, die von einigen Funktionsbausteinen ausgegeben werden.

Der 32-Bit-Fehler-Code besteht aus vier 8-Bit-Werten (DWORD).

4. Byte	3. Byte	2. Byte	1. Byte
Fehlerklasse	anwendungsspezifischer Fehler-Code	Fehlerquelle	Fehlerursache

7.2.1 Fehlerursache (1. Byte)

12336

Wert		Beschreibung
dez	hex	
0	00	keine Fehlerursache oder: anwendungsspezifischer Fehler
1	01	Bruch
2	02	Schluss
4	04	Überlast
5	05	Unterspannung
6	06	Überspannung
7	07	Stromregelung
8	08	Safety-Diagnose am Stromeingang
9	09	Safety-Diagnose am Spannungseingang
10	0A	Safety-Diagnose am aktivierten Ausgang ("stuck at 1", Querschluss)
11	0B	Safety-Diagnose am SafetySwitch
12	0C	Safety-Diagnose am Analog-Multiplexer
13	0D	Safety-Diagnose am deaktivierten Ausgang ("stuck at 1")
24	18	Temperatur
25	19	Relaiskontakt
26	20	Speichertest
27	21	Adresstest
48	30	Interruptsystem
49	31	Zeitbasis
50	32	Befehlsausführung
51	33	Ganzzahl-Überlauf oder: Division durch Null
56	38	FPU Underflow
57	39	FPU Overflow
58	3A	FPU Division durch Null
59	3B	FPU unspezifischer Fehler
112	70	Kommunikation zum Co-Prozessor
128	80	CRC
129	81	Daten korrupt
130	82	Speicherschutz
131	83	keine Daten
144	90	Watchdog
145	91	Trap
146	92	Safety-Core gestoppt
147	93	Assertion fehlgeschlagen
192	C0	CANsafety

Wert dez	hex	Beschreibung
194	C2	CAN Busoff
195	C3	CANsafety Empfangsfehler
196	C4	CANsafety Sendefehler
197	C5	CANsafety Konfiguration korrupt
224	E0	Board Link Warnung (ExtendedController)
225	E1	Board Link Fehler (ExtendedController)
240	F0	Seriennummer
241	F1	Laufzeitsystem abgelaufen
248	F8	falscher Parameter

7.2.2 Fehlerquelle (2. Byte)

12337

Wert dez	hex	Beschreibung
0	00	keine Fehlerquelle oder: anwendungsspezifischer Fehler
1	01	CPU
2	02	Peripherie-Prozessor
3	03	Co-Prozessor
4	04	Safety-Core
5	05	Safety-Core (Code)
6	06	Safety-Core (Trap)
8	08	Floating-Point-Unit
16...31	10...1F	Eingang 0...15 (Standard-Seite)
32...63	20...3F	Eingang 0...31 (Extended-Seite)
64...79	40...4F	Ausgang 0...15 (Standard-Seite)
80..111	50...6F	Ausgang 0...31 (Extended-Seite)
128...131	80...83	CAN 1...4
144	90	Relaisspannung VBBo (Standard-Seite)
145	91	Relaisspannung VBBr (Standard-Seite)
146	92	VBBo (Standard-Seite)
147	93	VBBr (Standard-Seite)
148	94	VBBs (Standard-Seite)
149	95	Klemme 15
150	96	Relaisspannung VBB1 (Extended-Seite)
151	97	Relaisspannung VBB2 (Extended-Seite)
152	98	Relaisspannung VBB3 (Extended-Seite)
153	99	Relaisspannung VBB4 (Extended-Seite)
154	9A	VBBRel (Extended-Seite)
155	9B	VBB1 (Extended-Seite)
156	9C	VBB2 (Extended-Seite)
157	9D	VBB3 (Extended-Seite)
158	9E	VBB4 (Extended-Seite)
160	A0	Analog-Multiplexer
161	A1	Analog-Referenz
176	B0	internes Flash
177	B1	externes Flash
178	B2	internes RAM
179	B3	externes RAM
192	C0	Code Startupper

Wert dez	hex	Beschreibung
193	C1	Code Bootloader
194	C2	Code Laufzeitsystem
195	C3	Daten Peripherie-Prozessor
196	C4	Bootprojekt
197	C5	Code Anwendungsprogramm
198	C6	Scratch-Pad RAM
199	C7	Code Peripherie-Prozessor
224	E0	Systemdaten
225	E1	Systemeinstellungen
226	E2	Systeminformation
227	E3	Kalibrierdaten
228	E4	FRAM / MRAM (Anwenderbereich)

7.2.3 Anwendungsspezifischer Fehler-Code (3. Byte)

12338

Bei einem anwendungsspezifischen Fehler ist vorgeschrieben:
ERRORCODE Byte 1 = Fehlerursache = 0x00
ERRORCODE Byte 2 = Fehlerquelle = 0x00

► Anwendungsspezifische Fehler mit ***ERROR_REPORT*** (→ Seite 382) der Steuerung melden.

Wert dez	Wert hex	Beschreibung
0	00	kein anwendungsspezifischer Fehler
> 0	> 00	anwendungsspezifischer Fehler

7.2.4 Fehlerklasse (4. Byte)

12339

Diese Angaben gelten nur, wenn Eingang TEST = FALSE.

Wert dez	Wert hex	Beschreibung
0	00	kein Fehler
1	01	allgemeiner Fehler, nicht sicherheitsrelevant > Fehlermerker, Fehler-Code > Fehler rücksetzen möglich
2	02	schwerer Fehler, Komponenten-Fehler > Fehlermerker, Fehler-Code > sicherer Zustand (SafetyController: Keep-Alive möglich) > Fehler rücksetzen möglich
3	03	kritischer Fehler > Fehler-Code > Anwendungsprogramm temporär unterbrochen > sicherer Zustand > Fehler rücksetzen möglich
4	04	fataler Fehler > Fehler-Code > Anwendungsprogramm abgebrochen > sicherer Zustand ► Power-Off-On-Reset erforderlich

14025

Fatale Fehler sind in der Anwendung nur dann sichtbar, wenn beim Auftreten des Fehlers der TEST-Eingang bereits aktiv ist.

Normalerweise führen fatale Fehler zum STOP der Steuerung, jedoch nicht bei aktivem TEST-Eingang.

7.2.5 Fehler-Codes: Beispiele

19274

Byte 2	▼ Byte 1 ▼		
Fehlerquelle [hex]	Fehlerursache [hex]	Beschreibung	Funktionsbaustein
10...1F	01	Leiterbruch Ixx	INPUT_ANALOG
20...3F (Ex)	01	Leiterbruch Ixx_E	INPUT_ANALOG_E
40...4F	01	Leiterbruch Qxx	
50...6F (Ex)	01	Leiterbruch Qxx_E	
10...1F	02	Kurzschluss Ixx	INPUT_ANALOG
20...3F (Ex)	02	Kurzschluss Ixx_E	INPUT_ANALOG_E
40...4F	02	Kurzschluss Qxx	
50...6F (Ex)	02	Kurzschluss Qxx_E	
10...1F	04	Überstrom Ixx	INPUT_ANALOG
20...3F (Ex)	04	Überstrom Ixx_E	INPUT_ANALOG_E
40...4F	04	Überlast Qxx	
50...6F (Ex)	04	Überlast Qxx_E	
90	05	Unterspannung Relaisspannung VBBs	
91	05	Unterspannung Relaisspannung VBBr	
92	05	Unterspannung VBBo	
93	05	Unterspannung VBBr	
94	05	Unterspannung VBBs	
95	05	Unterspannung Klemme 15	
96...99 (Ex)	05	Unterspannung Relaisspannung VBBx	
9A (Ex)	05	Unterspannung VBBrel	
9B...9E (Ex)	05	Unterspannung VBBx	
90	06	Überspannung Relaisspannung VBBs	
91	06	Überspannung Relaisspannung VBBr	
92	06	Überspannung VBBo	
93	06	Überspannung VBBr	
94	06	Überspannung VBBs > 32 V	
94	06	Überspannung VBBs > 34 V	
95	06	Überspannung Klemme 15	
96...99 (Ex)	06	Überspannung Relaisspannung VBBx	
9A (Ex)	06	Überspannung VBBrel	
9B...9E (Ex)	06	Überspannung VBBx	
40...4F	07	Stromregelung Qxx	
50...6F (Ex)	07	Stromregelung Qxx_E	
10...1F	08 (safe)	Safety-Diagnose am Stromeingang	
10...1F	09 (safe)	Safety-Diagnose am Spannungseingang	
40...4F	0A (safe)	Safety-Diagnose am aktivierten Ausgang ("stuck at 1", Querschluss)	
10...17	0B (safe)	Safety-Diagnose am SafetySwitch	SAFETY_SWITCH

Byte 2	▼ Byte 1 ▼		
Fehlerquelle [hex]	Fehlerursache [hex]	Beschreibung	Funktionsbaustein
A0	0B (safe)	Analogwerte Ueberwachung /Multiplexer	SAFETY_SWITCH
40...4F	0D (safe)	Safety-Diagnose am deaktivierten Ausgang ("stuck at 1")	
00	18	Temperaturfehler	
90...91	19 (safe)	Kontaktfehler Relais VBBo / VBBr	
B3	20	Speichertest im RAM fehlgeschlagen	
E4	20	Speichertest im FRAM/MRAM fehlgeschlagen	
B3	21 (safe)	Adresstest im RAM fehlgeschlagen	
01	30 (safe)	Falscher / fehlender Interrupt	
01	31 (safe)	Fehler Zeitbasis CPU	
08	39	Floating-Point Overflow	
08	3A	Floating-Point Division durch 0	
08	3B	unspezifizierter Floating-Point-Fehler	
C2	80	Prüfsummenfehler im LZS-Code (ifm-Code)	
C3 (safe)	80	Prüfsummenfehler im PCP-Daten-RAM	
C4	80	Prüfsummenfehler im Bootprojekt	
C5	80	Prüfsummenfehler im Anwendungsprogramm-Code	
C6	80	Prüfsummenfehler im SP-RAM	
C7 (safe)	80	Prüfsummenfehler im PCP-Code-RAM	
E0	80	Prüfsummenfehler in den Systemdaten	
E1	80	Prüfsummenfehler in den Systemvariablen	
E2	80	Prüfsummenfehler in den Systemparametern	
E3	80	Prüfsummenfehler in den Kalibrierdaten	
B3	81	Defekte Daten im RAM	
E3	83	Kalibrierdaten korrupt	
04	92 (safe)	Safety-Core gestoppt	
05	92 (safe)	Fehler Safety-Code	
80...83	C1	CANx Warning	
80...83	C2	CANx Busoff	
80...83	C3 (safe)	CANsafety Empfangsfehler	
80...83	C4 (safe)	CANsafety Sendefehler	
80, 82	C5 (safe)	CANsafety-Konfiguration korrupt	
00 (Ex)	E1	Board-Link-Fehler	
00	F0	Fehler Seriennummer	
00	F1	Laufzeitsystem abgelaufen	
00	F8	Parameterfehler	Alle FBs mit ERROR Ausgang
10...1F	F8	Parameterfehler Ixx	INPUT_ANALOG SET_INPUT_MODE
20...3F (Ex)	F8	Parameterfehler Ixx_E	INPUT_ANALOG_E SET_INPUT_MODE_E
40...4F	F8	Parameterfehler Qxx	OUTPUT_ANALOG SET_OUTPUT_MODE

Byte 2	▼ Byte 1 ▼		
Fehlerquelle [hex]	Fehlerursache [hex]	Beschreibung	Funktionsbaustein
50...6F (Ex)	F8	Parameterfehler Qxx_E	OUTPUT_ANALOG_E SET_OUTPUT_MODE_E
05 (safe)	ErrorCode	Fehler Safety-Code	
06 (safe)	TrapID	Fehler Safety-Code Trap	

Legende:
(Ex) = gilt nur für ExtendedController
(safe) = gilt nur für SafetyController

Die resultierende Fehlerklasse (= Byte 4) ergibt sich aus dem Zusammenhang der Situation und Parametrierung. Byte 3 (anwendungsspezifischer Fehler-Code) ist hier immer = 0.

7.3 Fehlermerker

Inhalt

14029
12373

In den folgenden Tabellen unterscheiden wir folgende Fehlermerker:

- Fehler der Ein- und Ausgänge
  - Kurzschluss
  - Leiterbruch
  - Überlastung
  - Stromregelung
- Fehler des Systems
  - Spannungsversorgung
  - Gerätetemperatur
  - Speicher
  - CPUs
  - Adressierung
  - Daten
- Fehler an den CAN-Schnittstellen

Wenn nicht anders angegeben, handelt es sich um "allgemeine Fehler". Allgemeine Fehler werden nur dem Anwendungsprogramm signalisiert. Es liegt in der Verantwortung des Programmierers, auf diese Fehler zu reagieren.

► Nach Beseitigen der Fehlerursache den Fehlermerker zurücksetzen mit FB ***ERROR_RESET*** (→ Seite 384).

> Bei Systemfehlern geht die Steuerung in der Regel in den STOP-Zustand.

Beim SafetyController gibt es zusätzlich...
• schwere Fehler
• fatale Fehler
Details → Kapitel ***Fehlerklassen*** (→ Seite 48)

7.3.1 Fehler der Eingänge (Standard-Seite)

14026

Fehlermeldung	Art	Beschreibung
ERROR_BREAK_Ix (x=0...n; Wert abhängig vom Gerät, → Datenblatt)	DWORD	Leiterbruch-Fehler an der Eingangsgruppe x Wenn Eingang = SAFETY ⇨ schwerer Fehler! [Bit 0 für Eingang 0] ... [Bit z für Eingang z] dieser Gruppe Bit = TRUE: Fehler Bit = FALSE: kein Fehler
ERROR_SHORT_Ix (x=0...n; Wert abhängig vom Gerät, → Datenblatt)	DWORD	Kurzschluss-Fehler an der Eingangsgruppe x Wenn Eingang = SAFETY ⇨ schwerer Fehler! [Bit 0 für Eingang 0] ... [Bit z für Eingang z] dieser Gruppe Bit = TRUE: Fehler Bit = FALSE: kein Fehler
ERROR_SAFETY_Ix (x=0...n; Wert abhängig vom Gerät, → Datenblatt)	DWORD	schwerer Fehler an der sicheren Eingangsgruppe x [Bit 0 für Eingang 0] ... [Bit z für Eingang z] dieser Gruppe Bit = TRUE: Fehler Bit = FALSE: kein Fehler
ERRORCODE	DWORD	Zuletzt eingetragener Fehler in der internen Fehlerliste Die Liste enthält alle aufgetretenen Fehler-Codes.

Die Fehlermeldung gilt als "schwerer Fehler", wenn der betreffende Eingang als SAFETY=TRUE konfiguriert wurde → Kapitel ***Schwere Fehler*** (→ Seite 49).

7.3.2 Fehler der Eingänge (Extended-Seite)

15789

Fehlermeldung	Art	Beschreibung
ERROR_BREAK_Ix_E (x=0...n; Wert abhängig vom Gerät, → Datenblatt)	DWORD	Leiterbruch-Fehler an der Extended-Eingangsgruppe x [Bit 0 für Eingang 0] ... [Bit z für Eingang z] dieser Gruppe Bit = TRUE: Fehler Bit = FALSE: kein Fehler
ERROR_CURRENT_Ix_E (x=0...n; Wert abhängig vom Gerät, → Datenblatt)	DWORD	Überstrom-Fehler an der Extended-Eingangsgruppe x [Bit 0 für Eingang 0] ... [Bit z für Eingang z] dieser Gruppe Bit = TRUE: Fehler Bit = FALSE: kein Fehler
ERROR_SHORT_Ix_E (x=0...n; Wert abhängig vom Gerät, → Datenblatt)	DWORD	Kurzschluss-Fehler an der Extended-Eingangsgruppe x [Bit 0 für Eingang 0] ... [Bit z für Eingang z] dieser Gruppe Bit = TRUE: Fehler Bit = FALSE: kein Fehler
ERRORCODE	DWORD	Zuletzt eingetragener Fehler in der internen Fehlerliste Die Liste enthält alle aufgetretenen Fehler-Codes.

7.3.3 Fehler der Ausgänge (Standard-Seite)

14030

Fehlermeldung	Art	Beschreibung
ERROR_BREAK_Qx (x=0...n; Wert abhängig vom Gerät, → Datenblatt)	DWORD	Leiterbruch-Fehler an der Ausgangsgruppe x Wenn Ausgang = SAFETY ⇨ schwerer Fehler! [Bit 0 für Ausgang 0] ... [Bit z für Ausgang z] dieser Gruppe Bit = TRUE: Fehler Bit = FALSE: kein Fehler
ERROR_CONTROL_Qx (x=0...n; Wert abhängig vom Gerät, → Datenblatt)	DWORD	Fehler Stromregelung an der Ausgangsgruppe x: Endwert kann nicht erreicht werden [Bit 0 für Ausgang 0] ... [Bit z für Ausgang z] dieser Gruppe Bit = TRUE: Fehler Bit = FALSE: kein Fehler
ERROR_OVERLOAD_Qx (x=0...n; Wert abhängig vom Gerät, → Datenblatt)	DWORD	Überlast-Fehler an der Ausgangsgruppe x [Bit 0 für Ausgang 0] ... [Bit z für Ausgang z] dieser Gruppe Bit = TRUE: Fehler Bit = FALSE: kein Fehler
ERROR_SHORT_Qx (x=0...n; Wert abhängig vom Gerät, → Datenblatt)	DWORD	Kurzschluss-Fehler an der Ausgangsgruppe x Wenn Ausgang = SAFETY ⇨ schwerer Fehler! [Bit 0 für Ausgang 0] ... [Bit z für Ausgang z] dieser Gruppe Bit = TRUE: Fehler Bit = FALSE: kein Fehler
ERROR_SAFETY_Qx (x=0...n; Wert abhängig vom Gerät, → Datenblatt)	DWORD	schwererFehler an der sicheren Ausgangsgruppe x [Bit 0 für Ausgang 0] ... [Bit z für Ausgang z] dieser Gruppe Bit = TRUE: Fehler Bit = FALSE: kein Fehler
ERRORCODE	DWORD	Zuletzt eingetragener Fehler in der internen Fehlerliste Die Liste enthält alle aufgetretenen Fehler-Codes.

Die Fehlermeldung gilt als "schwerer Fehler", wenn der betreffende Ausgang als SAFETY=TRUE konfiguriert wurde → Kapitel ***Schwere Fehler*** (→ Seite 49).

7.3.4 Fehler der Ausgänge (Extended-Seite)

15790

Fehlermeldung	Art	Beschreibung
ERROR_BREAK_Qx_E (x=0...n; Wert abhängig vom Gerät, → Datenblatt)	DWORD	Leiterbruch-Fehler an der Extended-Ausgangsgruppe x [Bit 0 für Ausgang 0] ... [Bit z für Ausgang z] dieser Gruppe Bit = TRUE: Fehler Bit = FALSE: kein Fehler
ERROR_CONTROL_Qx_E (x=0...n; Wert abhängig vom Gerät, → Datenblatt)	DWORD	Fehler Stromregelung an der Extended-Ausgangsgruppe x: Endwert kann nicht erreicht werden [Bit 0 für Ausgang 0] ... [Bit z für Ausgang z] dieser Gruppe Bit = TRUE: Fehler Bit = FALSE: kein Fehler
ERROR_OVERLOAD_Qx_E (x=0...n; Wert abhängig vom Gerät, → Datenblatt)	DWORD	Überlast-Fehler an der Extended-Ausgangsgruppe x [Bit 0 für Ausgang 0] ... [Bit z für Ausgang z] dieser Gruppe Bit = TRUE: Fehler Bit = FALSE: kein Fehler
ERROR_SHORT_Qx_E (x=0...n; Wert abhängig vom Gerät, → Datenblatt)	DWORD	Kurzschluss-Fehler an der Extended-Ausgangsgruppe x [Bit 0 für Ausgang 0] ... [Bit z für Ausgang z] dieser Gruppe Bit = TRUE: Fehler Bit = FALSE: kein Fehler
ERRORCODE	DWORD	Zuletzt eingetragener Fehler in der internen Fehlerliste Die Liste enthält alle aufgetretenen Fehler-Codes.

7.3.5 Fehler des Systems (Standard-Seite)

14027

Fehlermeldung	Art	Beschreibung
ERROR	BOOL	TRUE = Sammelfehlermeldung setzen, Relais ausschalten
ERROR_POWER	BOOL	Spannungs-Fehler für SUPPLY_VOLTAGE: TRUE: Wert außerhalb des zulässigen Bereichs > schwerer Fehler FALSE: Wert in Ordnung
ERROR_RELAIS	BOOL	Fehler Relais-Ansteuerung
ERROR_SYSTEM	BOOL	System-Fehler (nur sichtbar im TEST-Betrieb) TRUE: Geräte-Hardware defekt > Fataler Fehler ▶ Gerät an ifm senden FALSE: kein Fehler
ERROR_TEMPERATURE	BOOL	Temperatur-Fehler TRUE: Wert außerhalb des zulässigen Bereichs > fataler Fehler FALSE: Wert in Ordnung
ERROR_VBBx	BOOL	Versorgungsspannungs-Fehler an VBBx (x = O | R): TRUE: Wert außerhalb des zulässigen Bereichs > schwerer Fehler FALSE: Wert in Ordnung
ERRORCODE	DWORD	Zuletzt eingetragener Fehler in der internen Fehlerliste Die Liste enthält alle aufgetretenen Fehler-Codes.

7.3.6 Fehler des Systems (Extended-Seite)

15791

Fehlermeldung	Art	Beschreibung
BOARD_LINK_ERROR	BOOL	Die Verbindung zur Extended-Seite ist... TRUE: unterbrochen die Extended-Seite ist offline (!) Nach Unterbrechen der Verbindung ist keine automatische Neuverbindung möglich. ▶ Gerät neu starten! FALSE: in Ordnung
BOARD_LINK_WARNING	BOOL	Die Verbindung zur Extended-Seite ist... TRUE: gestört, aber noch funktionsfähig FALSE: in Ordnung
ERROR_IO_E	BOOL	Sammelfehlermeldung Ein-/Ausgangsfehler Extended-Seite TRUE: Fehler FALSE: kein Fehler
ERROR_POWER_E	BOOL	Spannungs-Fehler Extended-Seite: TRUE: Wert außerhalb des zulässigen Bereichs FALSE: Wert in Ordnung
ERROR_VBBx_E	BOOL	Versorgungsspannungs-Fehler an Extended-VBBx x = 1 | 2 | 3 | 4 TRUE: Wert außerhalb des zulässigen Bereichs FALSE: Wert in Ordnung
ERROR_VBBREL_E	BOOL	Versorgungsspannungs-Fehler an Relaisversorgung: TRUE: Wert außerhalb des zulässigen Bereichs FALSE: Wert in Ordnung
ERRORCODE	DWORD	Zuletzt eingetragener Fehler in der internen Fehlerliste Die Liste enthält alle aufgetretenen Fehler-Codes.

7.3.7 Fehler an den CAN-Schnittstellen

14028
12269

Wird der Mechanismus der sicheren Datenübertragung über den CAN-Bus gewählt (CANsafety), führen alle erkannten Fehler im Sender (Producer) und Empfänger (Consumer) der Daten zu einer Fehlermeldung "schwerer Fehler".
> CAN_SAFETY_ERROR_1 = TRUE
und / oder
> CAN_SAFETY_ERROR_2 = TRUE
Mit oder ohne CANsafety können CAN-Fehler über die CAN-Systemmerker überwacht werden.

Systemmerker (Symbolname)	Typ	Beschreibung
CANx_BUSOFF	BOOL	CAN-Schnittstelle x: Fehler "CAN-Bus off" Zurücksetzen des Fehler-Codes setzt auch den Merker zurück
CANx_ERRORCOUNTER_RX	BYTE	CAN-Schnittstelle x: Fehlerzähler Empfang Reset des Merkers ist via Schreibzugriff möglich
CANx_ERRORCOUNTER_TX	BYTE	CAN-Schnittstelle x: Fehlerzähler Versand Reset des Merkers ist via Schreibzugriff möglich
CANx_LASTERROR	BYTE	CAN-Schnittstelle x: Fehlernummer der letzten CAN-Übertragung (reserviert für ifm-interne Service-Zwecke)
CANx_WARNING	BOOL	CAN-Schnittstelle x: Warnschwelle erreicht (≥ 96) Reset des Merkers ist via Schreibzugriff möglich
CAN_SAFETY_ERROR_1	BOOL	Fehler bei den CANsafety-Nachrichten an CANsafety-Kanal 1 TRUE: schwerer Fehler aufgetreten FALSE: kein Fehler
CAN_SAFETY_ERROR_2	BOOL	Fehler bei den CANsafety-Nachrichten an CANsafety-Kanal 2 TRUE: schwerer Fehler aufgetreten FALSE: kein Fehler
ERRORCODE	DWORD	Zuletzt eingetragener Fehler in der internen Fehlerliste Die Liste enthält alle aufgetretenen Fehler-Codes.

x = 1...4 = Nummer der CAN-Schnittstelle

7.4 Reaktion auf System-Fehler

14033

ⓘ Für die sichere Verarbeitung der Daten im Anwendungsprogramm ist allein dessen Programmierer verantwortlich.

► Die spezifischen Fehlermerker und / oder Fehler-Codes im Anwendungsprogramm verarbeiten!
Über den Fehlermerker/Fehler-Code erhält man eine Fehlerbeschreibung.
Dieser Fehlermerker/Fehler-Code kann bei Bedarf weiter verarbeitet werden.

Nach der Analyse und Beseitigung der Fehler-Ursache:

► Grundsätzlich alle Fehlermerker durch das Anwendungsprogramm zurücksetzen.
Ohne ausdrückliches Rücksetzen der Fehlermerker bleiben die Merker gesetzt mit entsprechender Auswirkung im Anwendungsprogramm.

7.4.1 Relais: wichtige Hinweise!

14034

ACHTUNG
Vorzeitiger Verschleiß der Relaiskontakte möglich. ► Im Normalfall die Relais nur lastfrei schalten! Dazu via Anwendungsprogramm alle relevanten Ausgänge auf FALSE setzen!

7.5 Fehler-Codes konfigurieren und verwalten

12342

Hier zeigen wir Ihnen Funktionen, mit denen Sie Folgendes erreichen:
- anwendungsspezifische Fehler-Codes generieren
- Fehler-Codes auflisten oder löschen

ERROR_REPORT (→ Seite 382)	meldet dem System einen anwendungsspezifischen Fehler
ERROR_RESET (→ Seite 384)	setzt anstehende Fehlermeldungen zurück
PACK_ERRORCODE (→ Seite 388)	hilft beim Zusammenbauen eines ERRORCODE aus den Bytes für: • Fehlerklasse • anwendungsspezifischer Fehler • Fehlerquelle • Fehlerursache
SET_KEEP_ALIVE (→ Seite 386)	konfiguriert, welcher Ausgangskanal und welcher CANsafety-Kanal beim Auftreten eines bestimmten schweren Fehlers weiterbetrieben werden sollen, da sie von dem mit ERRORCODE gemeldeten Fehler unabhängig sind
SHOW_ERROR_LIST (→ Seite 389)	liest den aktuell vorliegenden Fehler-Code
UNPACK_ERRORCODE (→ Seite 390)	hilft beim Trennen eines ERRORCODE in die Bytes für: • Fehlerklasse • anwendungsspezifischer Fehler • Fehlerquelle • Fehlerursache

7.6 CAN / CANopen: Fehler und Fehlerbehandlung

Inhalt

1171

Die hier beschriebenen Fehlermechanismen werden von dem im Controller integrierten CAN-Controller automatisch abgearbeitet. Der Anwender hat darauf keinen Einfluss. Der Programmierer sollte (je nach Anwendung) auf gemeldete Fehler im Anwendungsprogramm reagieren.

Ziel der CAN-Fehler-Mechanismen ist es:

- Sicherstellung einheitlicher Datenobjekte im gesamten CAN-Netz
- Dauerhafte Funktionsfähigkeit des Netzes auch im Falle eines defekten CAN-Teilnehmers
- Unterscheidung zwischen zeitweiliger und dauerhafter Störung eines CAN-Teilnehmers
- Lokalisierung und Selbstabschaltung eines defekten Teilnehmers in 2 Stufen:
  • fehlerpassiv (error-passiv)
  • trennen vom Bus (bus-off)
  Dies ermöglicht einem zeitweilig gestörten Teilnehmer eine "Erholungspause".

Um dem interessierten Anwender einen Überblick über das Verhalten des CAN-Controllers im Fehlerfall zu geben, soll an dieser Stelle vereinfacht die Fehlerbehandlung beschrieben werden. Nach der Fehlererkennung werden die Informationen automatisch aufbereitet und stehen im Anwendungsprogramm dem Programmierer als CAN-Fehler-Bits zur Verfügung.

7.6.1 CAN-Fehler

8589

Fehlertelegramm

1172

Erkennt ein Busteilnehmer eine Fehlerbedingung, so sendet er sofort ein Fehlertelegramm und veranlasst damit den Abbruch der Übertragung bzw. das Verwerfen der von anderen Teilnehmern schon empfangenen fehlerfreien Nachrichten. Dadurch wird sichergestellt, dass allen Teilnehmern fehlerfreie und einheitliche Daten zur Verfügung stehen. Da das Fehlertelegramm unmittelbar übertragen wird, kann im Gegensatz zu anderen Feldbussystemen (diese warten eine festgelegte Quittierungszeit ab) sofort mit der Wiederholung der gestörten Nachricht durch den Absender begonnen werden. Dies ist eines der wichtigsten Merkmale von CAN.

Eine der grundsätzlichen Problematiken der seriellen Datenübertragung ist, dass ein dauerhaft gestörter oder defekter Busteilnehmer das gesamte System blockieren kann. Gerade die Fehlerbehandlung bei CAN würde solche Gefahr fördern. Um diesen Fall auszuschließen, ist ein Mechanismus erforderlich, welcher den Defekt eines Teilnehmers erkennt und diesen Teilnehmer gegebenenfalls vom Bus abschaltet.

Fehlerzähler

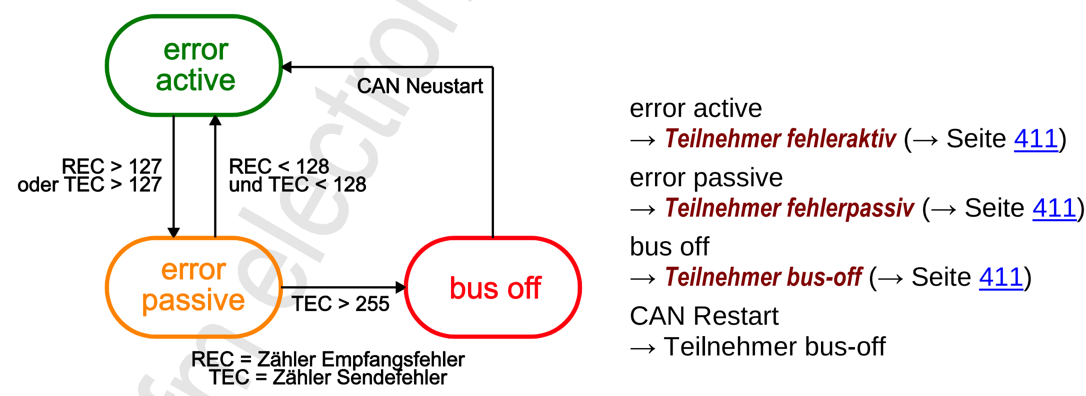
1173

Dazu sind im CAN-Controller ein Sende- und ein Empfangsfehlerzähler enthalten. Diese werden bei jedem fehlerhaften Sende- oder Empfangsvorgang heraufgezählt (inkrementiert). War eine Übertragung fehlerfrei, werden diese Zähler wieder heruntergezählt (dekrementiert).

Die Fehlerzähler werden jedoch im Fehlerfall stärker inkrementiert, als sie im Erfolgsfalle dekrementiert werden. Über eine bestimmte Zeitspanne kann dies zu einem merklichen Anstieg der Zählerstände führen, selbst wenn die Anzahl der ungestörten Nachrichten größer ist, als die Anzahl der gestörten Nachrichten. Längere fehlerfreie Zeitspannen bauen die Zählerstände langsam wieder ab. Die Zählerstände sind somit ein Maß für die relative Häufigkeit von gestörten Nachrichten.

Werden Fehler von einem Teilnehmer selbst als erster erkannt (= selbstverschuldete Fehler), wird bei diesem Teilnehmer der Fehler stärker "bestraft" als bei den anderen Busteilnehmern. Dazu wird der Zähler um einen höheren Betrag inkrementiert.

Übersteigt nun der Zählerstand eines Teilnehmers einen bestimmten Wert, kann davon ausgegangen werden, dass dieser Teilnehmer defekt ist. Damit dieser Teilnehmer den folgenden Busverkehr nicht weiter durch aktive Fehlermeldungen (error active) stört, wird er "fehlerpassiv" geschaltet (error passiv).



Grafik: Mechanismus des Fehlerzählers

error active
→ ***Teilnehmer fehleraktiv*** (→ Seite 411)

error passive
→ ***Teilnehmer fehlerpassiv*** (→ Seite 411)

bus off
→ ***Teilnehmer bus-off*** (→ Seite 411)

CAN Restart
→ Teilnehmer bus-off

Teilnehmer fehleraktiv

1174

Ein fehleraktiver Teilnehmer nimmt voll am Busverkehr teil und darf erkannte Fehler durch Senden des aktiven Fehlertelegramms signalisieren. Wie bereits beschrieben, wird dadurch die übertragene Nachricht zerstört.

Teilnehmer fehlerpassiv

1175

Ein fehlerpassiver Teilnehmer ist ebenfalls noch voll kommunikationsfähig. Er darf allerdings einen von ihm erkannten Fehler nur durch ein – den Busverkehr nicht störendes – passives Fehlerflag kenntlich machen. Ein fehlerpassiver Teilnehmer wird beim Unterschreiten eines festgelegten Zählerwertes wieder fehleraktiv.

Um den Anwender über das Ansteigen des Fehlerzählers zu informieren, wird bei einem Wert des Fehlerzählers > 96 die Systemvariable CANx_WARNING gesetzt. Der Teilnehmer ist in diesem Zustand noch fehleraktiv.

Teilnehmer bus-off

1176

Wird der Fehlerzählerwert weiter inkrementiert, wird nach Überschreiten eines Maximalzählerwertes der Teilnehmer vom Bus abgeschaltet (bus-off).

Um diesen Zustand anzuzeigen, wird im Anwendungsprogramm der Merker CANx_BUSOFF gesetzt.

> ! Der Fehler CANx_BUSOFF wird vom Laufzeitsystem automatisch behandelt und zurückgesetzt. Soll eine genauere Fehlerbehandlung und Auswertung über das Anwendungsprogramm erfolgen, muss ***CANx_ERRORHANDLER*** (→ Seite 208) eingesetzt werden. Der Fehler CANx_BUSOFF muss dann explizit durch das Anwendungsprogramm zurückgesetzt werden.

7.6.2 CANopen-Fehler

Inhalt

11670

Aufbau einer EMCY-Nachricht

Inhalt

8591

Die Signalisierung von Fehlerzuständen erfolgt unter CANopen über einen sehr einfachen, standardisierten Mechanismus. Jedes Auftreten eines Fehlers bei einem CANopen-Gerät wird über eine spezielle Nachricht signalisiert, die den Fehler genauer beschreibt.

Verschwindet ein Fehler oder seine Ursache nach einer bestimmten Zeit wieder, wird dieses Ereignis ebenfalls über die EMCY-Nachricht signalisiert. Die zuletzt aufgetretenen Fehler werden im Objektverzeichnis (Objekt 0x1003) abgelegt und können über einen SDO-Zugriff ausgelesen werden (→ ***CANx_SDO_READ*** (→ Seite 239)). Zusätzlich spiegelt sich die aktuelle Fehlersituation im Error-Register (Objekt 0x1001) wider.

Man unterscheidet folgende Fehler:

8046

Kommunikationsfehler

- Der CAN-Controller signalisiert CAN-Fehler.
(Das gehäufte Auftreten ist ein Indiz für physikalische Probleme. Diese Fehler können einen erheblichen Einfluss auf das Übertragungsverhalten und damit auf den Datendurchsatz eines Netzwerks haben.)
- Life-Guarding- oder Heartbeat-Fehler
- SYNC-Fehler (nur Slave)

Anwendungsfehler

- Kurzschluss oder Leiterbruch
- Temperatur zu hoch

Aufbau einer Fehlernachricht

8047

Eine Fehlernachricht (EMCY Message) hat folgenden Aufbau:

Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7	Byte 8
EMCY-Fehler-Code, wie im Objekt 0x1003 eingetragen		Objekt 0x1001	Herstellerspezifische Informationen				

Identifier

8048

Der Identifier für die Fehlernachricht besteht aus der Summe folgender Elemente:

EMCY-Default-Identifier 128 (0x80)
+
Node-ID

EMCY-Fehler-Code

8049

Er gibt detailliert Auskunft darüber, welcher Fehler aufgetreten ist. Eine Liste möglicher Fehler-Codes ist bereits im Kommunikationsprofil definiert. Fehler-Codes, die nur für eine bestimmte Geräteklasse gültig sind, werden im jeweiligen Geräteprofil dieser Geräteklasse festgelegt.

Objekt 0x1003 (Error Field)

8050

Das Objekt 0x1003 stellt den Fehlerspeicher eines Gerätes dar. Die Subindizes enthalten die zuletzt aufgetretenen Fehler, die ein Fehler-Telegramm ausgelöst haben.

Tritt ein neuer Fehler auf, dann wird sein EMCY-Fehler-Code immer im Subindex 0x1 gespeichert. Alle anderen, älteren Fehler werden im Fehlerspeicher um einen Platz nach hinten geschoben, also der Subindex um 1 erhöht. Falls alle unterstützten Subindizes belegt sind, wird der älteste Fehler gelöscht. Der Subindex 0x0 wird auf die Anzahl der gespeicherten Fehler erhöht. Nachdem alle Fehler behoben sind, wird in das Fehlerfeld des Subindex 0x1 der Wert "0" geschrieben.

Um den Fehlerspeicher zu löschen, kann der Subindex 0x0 mit dem Wert "0" beschrieben werden. Andere Werte dürfen nicht eingetragen werden.

Gerätefehler signalisieren

1880

Wie beschrieben, werden EMCY-Nachrichten versendet, wenn Fehler in einem Gerät auftreten. Im Unterschied zu frei programmierbaren Geräten, werden beispielsweise von dezentralen Ein-/Ausgangsmodulen (z.B. CompactModule CR2033) Fehlermeldungen automatisch verschickt. Entsprechende Fehler-Codes → jeweiliges Gerätehandbuch.

Die programmierbaren Geräte erzeugen nur dann automatisch eine EMCY-Nachricht (z.B. für "Kurzschluss am Ausgang Q07"), wenn einer der folgenden FBs in das Anwendungsprogramm eingebunden wird:

CANx_MASTER_EMCY_HANDLER (→ Seite 221)	verwaltet den geräteeigenen Fehlerstatus des CANopen-Masters an der CAN-Schnittstelle x x = 1...n = Nummer der CAN-Schnittstelle (je nach Gerät, → Datenblatt)
CANx_SLAVE_EMCY_HANDLER (→ Seite 231)	verwaltet den geräteeigenen Fehlerstatus des CANopen-Slaves an der CAN-Schnittstelle x: • Error Register (Index 0x1001) und • Error Field (Index 0x1003) des CANopen Objektverzeichnis x = 1...n = Nummer der CAN-Schnittstelle (je nach Gerät, → Datenblatt)

Übersicht der automatisch verschickten EMCY-Fehler-Codes für alle mit CODESYS programmierbaren **ecomat*mobile***-Geräte → Kapitel ***Übersicht CANopen-Error-Codes*** (→ Seite 415).

Sollen zusätzlich noch anwendungsspezifische Fehler durch das Anwendungsprogramm verschickt werden, dann einen der folgenden FBs einsetzen:

CANx_MASTER_SEND_EMERGENCY (→ Seite 222)	versendet anwendungsspezifische Fehlerstatus des CANopen-Masters an der CAN-Schnittstelle x x = 1...n = Nummer der CAN-Schnittstelle (je nach Gerät, → Datenblatt)
CANx_SLAVE_SEND_EMERGENCY (→ Seite 233)	versendet anwendungsspezifische Fehlerstatus des CANopen-Slaves an der CAN-Schnittstelle x x = 1...n = Nummer der CAN-Schnittstelle (je nach Gerät, → Datenblatt)

Übersicht CANopen-Error-Codes

8545

Error Code (hex)	Meaning / Bedeutung
00xx	Reset or no Error (Fehler rücksetzen / kein Fehler)
10xx	Generic Error (allgemeiner Fehler)
20xx	Current (Stromfehler)
21xx	Current, device input side (Stromfehler, eingangsseitig)
22xx	Current inside the device (Stromfehler im Geräteinnern)
23xx	Current, device output side (Stromfehler, ausgangsseitig)
30xx	Voltage (Spannungsfehler)
31xx	Mains Voltage
32xx	Voltage inside the device (Spannungsfehler im Geräteinnern)
33xx	Output Voltage (Spannungsfehler, ausgangsseitig)
40xx	Temperature (Temperaturfehler)
41xx	Ambient Temperature (Umgebungstemperaturfehler)
42xx	Device Temperature (Gerätetemperaturfehler)
50xx	Device Hardware (Geräte-Hardware-Fehler)
60xx	Device Software (Geräte-Software-Fehler)
61xx	Internal Software (Firmware-Fehler)
62xx	User Software (Applications-Software)
63xx	Data Set (Daten-/Parameterfehler)
70xx	Additional Modules (zusätzliche Module)
80xx	Monitoring (Überwachung)
81xx	Communication (Kommunikation)
8110	CAN Overrun-objects lost (CAN Überlauf-Datenverlust)
8120	CAN in Error Passiv Mode (CAN im Modus "fehlerpassiv")
8130	Life Guard Error or Heartbeat Error (Guarding-Fehler oder Heartbeat-Fehler)
8140	Recovered from Bus off (Bus-Off zurückgesetzt)
8150	Transmit COB-ID collision (Senden "Kollision des COB-ID")
82xx	Protocol Error (Protokollfehler)
8210	PDO not processed due to length error (PDO nicht verarbeitet, fehlerhafte Längenangabe)
8220	PDO length exceeded (PDO Längenfehler, ausgangsseitig)
90xx	External Error (Externer Fehler)
F0xx	Additional Functions (zusätzliche Funktionen)
FFxx	Device specific (gerätespezifisch)

Objekt 0x1001 (Error-Register)

8547

Dieses Objekt spiegelt den allgemeinen Fehlerzustand eines CANopen-Gerätes wider. Das Gerät ist dann als fehlerfrei anzusehen, wenn das Objekt 0x1001 keinen Fehler mehr signalisiert.

Bit	Meaning (Bedeutung)
0	Generic Error (allgemeiner Fehler)
1	Current (Stromfehler)
2	Voltage (Spannungsfehler)
3	Temperature (Temperaturfehler)
4	Communication Error (Kommunikationsfehler)
5	Device Profile specific (Geräteprofil spezifisch)
6	Reserved – always 0 (reserviert – immer 0)
7	manufacturer specific (herstellerspezifisch)

Für eine Fehlermeldung können mehrere Bits im Error-Register gleichzeitig gesetzt sein.

Beispiel: CR2033, Meldung "Leiterbruch" an Kanal 2 (→ Installationsanleitung des Geräts):

COB-ID	DLC	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7	Byte 8
0x80 + Node-ID		**00**	**FF**	**81**	**10**	**00**	00	00	00

Error-Code = 0xFF00

Error-Register = 0x81 = 0b1000 0001, besteht also aus folgenden Fehlern:

- generic error (allgemeiner Fehler)
- manufacturer specific (herstellerspezifisch)

Betroffener Kanal = 0x0010 = 0b0000 0000 0001 0000 = Kanal 2

Herstellerspezifische Informationen

8548

Hier kann ein Gerätehersteller zusätzliche Fehlerinformationen mitteilen. Das Format ist dabei frei wählbar.

Beispiel:

In einem Gerät treten zwei Fehler auf und werden über den Bus gemeldet:

- Kurzschluss der Ausgänge:
  Fehler-Code 0x2308,
  im Objekt 0x1001 wird der Wert 0x03 (0b0000 0011) eingetragen
  (allg. Fehler und Stromfehler)
- CAN-Überlauf:
  Fehler-Code 0x8110,
  im Objekt 0x1001 wird der Wert 0x13 (0b0001 0011) eingetragen
  (allg. Fehler, Stromfehler und Kommunikationsfehler)

>> CAN-Überlauf bearbeitet:
  Fehler-Code 0x0000,
  im Objekt 0x1001 wird der Wert 0x03 (0b0000 0011) eingetragen
  (allg. Fehler, Stromfehler, Kommunikationsfehler zurückgesetzt.)

Nur aus dieser Information kann man entnehmen, dass der Kommunikationsfehler nicht mehr anliegt.

Übersicht CANopen-EMCY-Codes (Standard-Seite)

13102

Die folgenden EMCY-Meldungen werden automatisch versendet, wenn der FB ***CANx_MASTER_EMCY_HANDLER*** (→ Seite 221) zyklisch aufgerufen wird.

EMCY-Code Objekt 0x1003		Objekt 0x1001	herstellerspezifische Informationen					
Byte 0 [hex]	Byte 1 [hex]	Byte 2 [hex]	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7	Beschreibung
00	21	03	I07...I00	I15...I08				Leiterbruch Eingänge
08	21	03	I07...I00	I15...I08				Kurzschluss Eingänge
10	21	03	I07...I00	I15...I08				Überstrom 0...20 mA
00	23	03	Q07...Q00	Q15...Q08				Leiterbruch Ausgänge
08	23	03	Q07...Q00	Q15...Q08				Kurzschluss Ausgänge
10	23	03	Q07...Q00	Q15...Q08				Überlast Ausgänge
00	31	05						Versorgungsspannung VBBs
00	33	05						Ausgangsspannung VBBo
08	33	05						Ausgangsspannung VBBr
00	42	09						Übertemperatur

13094

Die Angaben für CANx gelten für jede der CAN-Schnittstellen.

EMCY-Code Objekt 0x1003		Objekt 0x1001	herstellerspezifische Informationen					
Byte 0 [hex]	Byte 1 [hex]	Byte 2 [hex]	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7	Beschreibung
00	80	11	---	---	---	---	---	CANx Monitoring SYNC-Error (nur Slave)
00	81	11	---	---	---	---	---	CANx Warngrenze (> 96)
10	81	11	---	---	---	---	---	CANx Empfangspuffer Überlauf
11	81	11	---	---	---	---	---	CANx Sendepuffer Überlauf
30	81	11	---	---	---	---	---	CANx Guard-/Heartbeat-Error (nur Slave)

Übersicht CANopen-EMCY-Codes (Extended-Seite)

13103

Die folgenden EMCY-Meldungen werden automatisch versendet, wenn der FB ***CANx_MASTER_EMCY_HANDLER*** (→ Seite 221) zyklisch aufgerufen wird.

EMCY-Code Objekt 0x1003		Objekt 0x1001	herstellerspezifische Informationen					
Byte 0 [hex]	Byte 1 [hex]	Byte 2 [hex]	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7	Beschreibung
01	21	03	I07_E ...I00_E	I15_E ...I08_E				Leiterbruch Eingänge
09	21	03	I07_E ...I00_E	I15_E ...I08_E				Kurzschluss Eingänge
11	21	03	I07_E ...I00_E	I15_E ...I08_E				Überstrom 0…20 mA
01	23	03	Q07_E ...Q00_E	Q15_E ...Q08_E	Q23_E ...Q16_E	Q31_E ...Q24_E		Leiterbruch Ausgänge
09	23	03	Q07_E ...Q00_E	Q15_E ...Q08_E	Q23_E ...Q16_E	Q31_E ...Q24_E		Kurzschluss Ausgänge
10	23	03	Q07_E ...Q00_E	Q15_E ...Q08_E				Überlast Ausgänge
10	33	05						Ausgangsspannung VBB1
11	33	05						Ausgangsspannung VBB2
12	33	05						Ausgangsspannung VBB3
13	33	05						Ausgangsspannung VBB4
18	33	05						Versorgung Relais

8 Anhang

Inhalt

1664

Hier stellen wir Ihnen – ergänzend zu den Angaben in den Datenblättern – zusammenfassende Tabellen zur Verfügung.

8.1 Systemmerker

Inhalt

12167

> ! Für die Programmierung sollten nur die Symbolnamen genutzt werden, da sich die zugehörigen Merkeradressen bei einer Erweiterung der Steuerungskonfiguration ändern können.

(→ Kapitel ***Fehler-Codes und Diagnoseinformationen*** (→ Seite 391))

8.1.1 Systemmerker: CAN

12814

Systemmerker (Symbolname)	Typ	Beschreibung
CANx_BAUDRATE	WORD	CAN-Schnittstelle x: eingestellte Baudrate in [kBaud]
CANx_BUSOFF	BOOL	CAN-Schnittstelle x: Fehler "CAN-Bus off" Zurücksetzen des Fehler-Codes setzt auch den Merker zurück
CANx_DOWNLOADID	BYTE	CAN-Schnittstelle x: eingestellter Download-Identifier
CANx_ERRORCOUNTER_RX	BYTE	CAN-Schnittstelle x: Fehlerzähler Empfang Reset des Merkers ist via Schreibzugriff möglich
CANx_ERRORCOUNTER_TX	BYTE	CAN-Schnittstelle x: Fehlerzähler Versand Reset des Merkers ist via Schreibzugriff möglich
CANx_LASTERROR	BYTE	CAN-Schnittstelle x: Fehlernummer der letzten CAN-Übertragung (reserviert für ifm-interne Service-Zwecke)
CANx_WARNING	BOOL	CAN-Schnittstelle x: Warnschwelle erreicht (≥ 96) Reset des Merkers ist via Schreibzugriff möglich
CAN_SAFETY_ERROR_1	BOOL	Fehler bei den CANsafety-Nachrichten an CANsafety-Kanal 1 TRUE: schwerer Fehler aufgetreten FALSE: kein Fehler
CAN_SAFETY_ERROR_2	BOOL	Fehler bei den CANsafety-Nachrichten an CANsafety-Kanal 2 TRUE: schwerer Fehler aufgetreten FALSE: kein Fehler

x = 1...4 = Nummer der CAN-Schnittstelle

8.1.2 Systemmerker: SAE-J1939

12815

Systemmerker (Symbolname)	Typ	Beschreibung
J1939_RECEIVE_OVERWRITE	BOOL	Einstellung gilt nur für J1939 Daten, die nicht über ein J1939-Transportprotokoll übertragen wurden. TRUE: Alte Daten werden durch die neuen Daten überschrieben, wenn die alten Daten noch nicht aus der Funktionsbaustein-Instanz ausgelesen wurden FALSE: Neue Daten werden verworfen, solange die alten Daten noch nicht aus der Funktionsbaustein-Instanz ausgelesen wurden Neue Daten können eintreffen, bevor die alten ausgelesen wurden, wenn der IEC-Zyklus länger ist als die Aktualisierungsfrequenz der J1939-Daten
J1939_TASK	BOOL	TRUE: J1939-Task ist aktiv (= Initialwert) FALSE: J1939-Task ist nicht aktiv

8.1.3 Systemmerker: Fehlermerker (Standard-Seite)

12816

Systemmerker (Symbolname)	Typ	Beschreibung
ERROR	BOOL	TRUE: sicherer Zustand eingenommen FALSE: kein schwerer Fehler aufgetreten
ERROR_BREAK_Ix (x=0...n; Wert abhängig vom Gerät, → Datenblatt)	DWORD	Leiterbruch-Fehler an der Eingangsgruppe x Wenn Eingang = SAFETY ⇨ schwerer Fehler! [Bit 0 für Eingang 0] ... [Bit z für Eingang z] dieser Gruppe Bit = TRUE: Fehler Bit = FALSE: kein Fehler
ERROR_BREAK_Qx (x=0...n; Wert abhängig vom Gerät, → Datenblatt)	DWORD	Leiterbruch-Fehler an der Ausgangsgruppe x Wenn Ausgang = SAFETY ⇨ schwerer Fehler! [Bit 0 für Ausgang 0] ... [Bit z für Ausgang z] dieser Gruppe Bit = TRUE: Fehler Bit = FALSE: kein Fehler
ERROR_CONTROL_Qx (x=0...n; Wert abhängig vom Gerät, → Datenblatt)	DWORD	Fehler Stromregelung an der Ausgangsgruppe x: Endwert kann nicht erreicht werden [Bit 0 für Ausgang 0] ... [Bit z für Ausgang z] dieser Gruppe Bit = TRUE: Fehler Bit = FALSE: kein Fehler
ERROR_CURRENT_Ix (x=0...n; Wert abhängig vom Gerät, → Datenblatt)	DWORD	Überstrom-Fehler an der Eingangsgruppe x Wenn Eingang = SAFETY ⇨ schwerer Fehler! [Bit 0 für Eingang 0] ... [Bit z für Eingang z] dieser Gruppe Bit = TRUE: Fehler Bit = FALSE: kein Fehler
ERROR_OVERLOAD_Qx (x=0...n; Wert abhängig vom Gerät, → Datenblatt)	DWORD	Überlast-Fehler an der Ausgangsgruppe x Wenn Ausgang = SAFETY ⇨ schwerer Fehler! [Bit 0 für Ausgang 0] ... [Bit z für Ausgang z] dieser Gruppe Bit = TRUE: Fehler Bit = FALSE: kein Fehler
ERROR_POWER	BOOL	Spannungs-Fehler für SUPPLY_VOLTAGE: TRUE: Wert außerhalb des zulässigen Bereichs > schwerer Fehler FALSE: Wert in Ordnung
ERROR_SAFETY_Ix (x=0...n; Wert abhängig vom Gerät, → Datenblatt)	DWORD	schwerer Fehler an der sicheren Eingangsgruppe x [Bit 0 für Eingang 0] ... [Bit z für Eingang z] dieser Gruppe Bit = TRUE: Fehler Bit = FALSE: kein Fehler
ERROR_SAFETY_Qx (x=0...n; Wert abhängig vom Gerät, → Datenblatt)	DWORD	schwererFehler an der sicheren Ausgangsgruppe x [Bit 0 für Ausgang 0] ... [Bit z für Ausgang z] dieser Gruppe Bit = TRUE: Fehler Bit = FALSE: kein Fehler
ERROR_SHORT_Ix (x=0...n; Wert abhängig vom Gerät, → Datenblatt)	DWORD	Kurzschluss-Fehler an der Eingangsgruppe x Wenn Eingang = SAFETY ⇨ schwerer Fehler! [Bit 0 für Eingang 0] ... [Bit z für Eingang z] dieser Gruppe Bit = TRUE: Fehler Bit = FALSE: kein Fehler
ERROR_SHORT_Qx (x=0...n; Wert abhängig vom Gerät, → Datenblatt)	DWORD	Kurzschluss-Fehler an der Ausgangsgruppe x Wenn Ausgang = SAFETY ⇨ schwerer Fehler! [Bit 0 für Ausgang 0] ... [Bit z für Ausgang z] dieser Gruppe Bit = TRUE: Fehler Bit = FALSE: kein Fehler
ERROR_SYSTEM	BOOL	System-Fehler (nur sichtbar im TEST-Betrieb) TRUE: Geräte-Hardware defekt > Fataler Fehler ▶ Gerät an ifm senden FALSE: kein Fehler

Systemmerker (Symbolname)	Typ	Beschreibung
ERROR_TEMPERATURE	BOOL	Temperatur-Fehler TRUE: Wert außerhalb des zulässigen Bereichs > fataler Fehler FALSE: Wert in Ordnung
ERROR_VBBx	BOOL	Versorgungsspannungs-Fehler an VBBx (x = O | R): TRUE: Wert außerhalb des zulässigen Bereichs > schwerer Fehler FALSE: Wert in Ordnung
ERRORCODE	DWORD	Zuletzt eingetragener Fehler in der internen Fehlerliste Die Liste enthält alle aufgetretenen Fehler-Codes.
LAST_RESET	BYTE	Grund für den letzten Reset: 01 = PowerOn-Reset 03 = unbekannter Grund

8.1.4 Systemmerker: Fehlermerker (Extended-Seite)

13104

Systemmerker (Symbolname)	Typ	Beschreibung
BOARD_LINK_WARNING	BOOL	Die Verbindung zur Extended-Seite ist... TRUE: gestört, aber noch funktionsfähig FALSE: in Ordnung
BOARD_LINK_ERROR	BOOL	Die Verbindung zur Extended-Seite ist... TRUE: unterbrochen die Extended-Seite ist offline (!) Nach Unterbrechen der Verbindung ist keine automatische Neuverbindung möglich. ► Gerät neu starten! FALSE: in Ordnung
ERROR_BREAK_Ix_E (x=0...n; Wert abhängig vom Gerät, → Datenblatt)	DWORD	Leiterbruch-Fehler an der Extended-Eingangsgruppe x [Bit 0 für Eingang 0] ... [Bit z für Eingang z] dieser Gruppe Bit = TRUE: Fehler Bit = FALSE: kein Fehler
ERROR_BREAK_Qx_E (x=0...n; Wert abhängig vom Gerät, → Datenblatt)	DWORD	Leiterbruch-Fehler an der Extended-Ausgangsgruppe x [Bit 0 für Ausgang 0] ... [Bit z für Ausgang z] dieser Gruppe Bit = TRUE: Fehler Bit = FALSE: kein Fehler
ERROR_CONTROL_Qx_E (x=0...n; Wert abhängig vom Gerät, → Datenblatt)	DWORD	Fehler Stromregelung an der Extended-Ausgangsgruppe x: Endwert kann nicht erreicht werden [Bit 0 für Ausgang 0] ... [Bit z für Ausgang z] dieser Gruppe Bit = TRUE: Fehler Bit = FALSE: kein Fehler
ERROR_CURRENT_Ix_E (x=0...n; Wert abhängig vom Gerät, → Datenblatt)	DWORD	Überstrom-Fehler an der Extended-Eingangsgruppe x [Bit 0 für Eingang 0] ... [Bit z für Eingang z] dieser Gruppe Bit = TRUE: Fehler Bit = FALSE: kein Fehler
ERROR_OVERLOAD_Qx_E (x=0...n; Wert abhängig vom Gerät, → Datenblatt)	DWORD	Überlast-Fehler an der Extended-Ausgangsgruppe x [Bit 0 für Ausgang 0] ... [Bit z für Ausgang z] dieser Gruppe Bit = TRUE: Fehler Bit = FALSE: kein Fehler
ERROR_SHORT_Ix_E (x=0...n; Wert abhängig vom Gerät, → Datenblatt)	DWORD	Kurzschluss-Fehler an der Extended-Eingangsgruppe x [Bit 0 für Eingang 0] ... [Bit z für Eingang z] dieser Gruppe Bit = TRUE: Fehler Bit = FALSE: kein Fehler
ERROR_SHORT_Qx_E (x=0...n; Wert abhängig vom Gerät, → Datenblatt)	DWORD	Kurzschluss-Fehler an der Extended-Ausgangsgruppe x [Bit 0 für Ausgang 0] ... [Bit z für Ausgang z] dieser Gruppe Bit = TRUE: Fehler Bit = FALSE: kein Fehler
ERROR_VBBx_E	BOOL	Versorgungsspannungs-Fehler an Extended-VBBx x = 1 | 2 | 3 | 4 TRUE: Wert außerhalb des zulässigen Bereichs FALSE: Wert in Ordnung
ERROR_VBBREL_E	BOOL	Versorgungsspannungs-Fehler an Relaisversorgung: TRUE: Wert außerhalb des zulässigen Bereichs FALSE: Wert in Ordnung

8.1.5 Systemmerker: LED (Standard-Seite)

12817

Systemmerker (Symbolname)	Typ	Beschreibung
LED	WORD	LED-Farbe für "LED eingeschaltet": 0x0000 = LED_GREEN (voreingestellt) 0x0001 = LED_BLUE 0x0002 = LED_RED 0x0003 = LED_WHITE 0x0004 = LED_BLACK 0x0005 = LED_MAGENTA 0x0006 = LED_CYAN 0x0007 = LED_YELLOW
LED_X	WORD	LED-Farbe für "LED ausgeschaltet": 0x0000 = LED_GREEN 0x0001 = LED_BLUE 0x0002 = LED_RED 0x0003 = LED_WHITE 0x0004 = LED_BLACK (voreingestellt) 0x0005 = LED_MAGENTA 0x0006 = LED_CYAN 0x0007 = LED_YELLOW
LED_MODE	WORD	LED-Blinkfrequenz: 0x0000 = LED_2HZ (blinkt mit 2 Hz; voreingestellt) 0x0001 = LED_1HZ (blinkt mit 1 Hz) 0x0002 = LED_05HZ (blinkt mit 0,5 Hz) 0x0003 = LED_0HZ (leuchtet dauernd mit Wert in LED)

8.1.6 Systemmerker: LED (Extended-Seite)

12824

Systemmerker (Symbolname)	Typ	Beschreibung
LED_E	WORD	LED-Farbe für "LED eingeschaltet": 0x0000 = LED_GREEN (voreingestellt) 0x0001 = LED_BLUE 0x0002 = LED_RED 0x0003 = LED_WHITE 0x0004 = LED_BLACK 0x0005 = LED_MAGENTA 0x0006 = LED_CYAN 0x0007 = LED_YELLOW
LED_X_E	WORD	LED-Farbe für "LED ausgeschaltet": 0x0000 = LED_GREEN 0x0001 = LED_BLUE 0x0002 = LED_RED 0x0003 = LED_WHITE 0x0004 = LED_BLACK (voreingestellt) 0x0005 = LED_MAGENTA 0x0006 = LED_CYAN 0x0007 = LED_YELLOW
LED_MODE_E	WORD	LED-Blinkfrequenz: 0x0000 = LED_2HZ (blinkt mit 2 Hz; voreingestellt) 0x0001 = LED_1HZ (blinkt mit 1 Hz) 0x0002 = LED_05HZ (blinkt mit 0,5 Hz) 0x0003 = LED_0HZ (leuchtet dauernd mit Wert in LED_E)

8.1.7 Systemmerker: Spannungen (Standard-Seite)

12818

Systemmerker (Symbolname)	Typ	Beschreibung
CLAMP_15_VOLTAGE	WORD	Spannung an Klemme 15 in [mV]
REF_VOLTAGE	WORD	Spannung am Referenzspannungsausgang in [mV]
REFERENCE_VOLTAGE_5	BOOL	Referenzspannungsausgang mit 5 V aktiviert
REFERENCE_VOLTAGE_10	BOOL	Referenzspannungsausgang mit 10 V aktiviert
RELAIS_VBBy y = O | R	BOOL	TRUE: Relais für VBBy aktiviert Ausgangsgruppe x wird mit Spannung versorgt (x = 1 | 2) FALSE: Relais für VBBy ausgeschaltet Ausgangsgruppe x ist spannungslos
SERIAL_MODE	BOOL	serielle Schnittstelle (RS232) für die Verwendung in der Anwendung aktivieren TRUE: RS232-Schnittstelle kann in der Anwendung verwendet werden, jedoch nicht mehr zum Programmieren, Debuggen oder Monitoren des Geräts. FALSE: RS232-Schnittstelle kann in der Anwendung nicht verwendet werden. Programmieren, Debuggen oder Monitoren des Geräts ist möglich.
SUPPLY_SWITCH	BOOL	Bit zum Abschalten der Versorgungs-Selbsthaltung VBBs. Das Rücksetzen des Merkers wird vom Laufzeitsystem nur akzeptiert, wenn die Spannung an Klemme 15 < 4 V ist, ansonsten wird der Merker wieder aktiviert. Die Trennung von VBBs erfolgt vor dem Beginn des nächsten SPS-Zyklus. Abhängig vom Ladezustand der internen Kondensatoren kann es noch eine gewisse Zeit dauern, bis das Gerät abschaltet. TRUE: Versorgung des Geräts über VBBs ist aktiv FALSE: Versorgung des Geräts über VBBs wird deaktiviert
SUPPLY_VOLTAGE	WORD	Versorgungsspannung an VBBs in [mV]
TEST	BOOL	TRUE: Test-Eingang ist aktiv: • Programmiermodus ist freigeben • Software-Download ist möglich • die sicheren Ausgänge sind deaktiviert • es werden keine CANsafety-Nachrichten versendet • Zustand des Anwendungsprogramms ist abfragbar • kein Schutz der gespeicherten Software möglich FALSE: laufender Betrieb der Anwendung
VBBx_RELAIS_VOLTAGE x = O | R	WORD	Versorgungsspannung an VBBx nach Relaiskontakt in [mV]
VBBx_VOLTAGE x = O | R	WORD	Versorgungsspannung an VBBx in [mV]

8.1.8 Systemmerker: Spannungen (Extended-Seite)

13109

Systemmerker (Symbolname)	Typ	Beschreibung
RELAIS_VBBy_E y = O | R	BOOL	TRUE: Relais für VBBy aktiviert VBBo → VBB1 + VBB2 VBBr → VBB2 + VBB4 Ausgangsgruppe x wird mit Spannung versorgt (x = 1 | 2 | 3 | 4) FALSE: Relais für VBBy ausgeschaltet Ausgangsgruppe x ist spannungslos
VBBx_E x = 1 | 2 | 3 | 4	WORD	Versorgungsspannung an VBBx_E in [mV]
VBBx_RELAIS_VOLTAGE_E x = 1 | 2 | 3 | 4	WORD	Versorgungsspannung an VBBx nach Relaiskontakt in [mV]
VBB_RELAIS_VOLTAGE_E	WORD	Versorgungsspannung für Relaisversorgung in [mV]

8.1.9 Systemmerker: 16 Eingänge und 16 Ausgänge (Standard-Seite)

12793

Systemmerker (Symbolname)	Typ	Beschreibung
ANALOGxx xx = 00...15	WORD	Analog-Eingang xx: gefilterter A/D-Wandler-Rohwert (12 Bit) ohne Kalibrierung und Normierung
CURRENTxx xx = 00...15	WORD	Analog-Ausgang xx: Gefilterte A/D-Wandler-Rohwerte (12 Bit) der Strommessung ohne Kalibrierung und Normierung
Ixx xx = 00...15	BOOL	Status am Binäreingang xx Voraussetzung: Eingang ist als Binäreingang konfiguriert (MODE = IN_DIGITAL_H oder IN_DIGITAL_L) TRUE: Spannung am Binäreingang > 70 % von VBBS FALSE: Spannung am Binäreingang < 30 % von VBBS oder: nicht als Binäreingang konfiguriert oder: falsch konfiguriert
Ixx_DFILTER xx = 00...11	DWORD	Impulseingang xx: Impulsdauer in [µs], die als Glitch ignoriert werden soll. Die Erfassung des Eingangssignals verzögert sich um die eingestellte Zeit. zugelassen = 0...100 000 µs voreingestellt = 0 µs = kein Filter
Ixx_FILTER xx = 00...15	BYTE:=4	Binär- und Analogeingang xx: Grenzfrequenz (oder Signalanstiegszeit) des Software-Tiefpass-Filters erster Ordnung 0 = 0x00 = kein Filter 1 = 0x01 = 390 Hz (1 ms) 2 = 0x02 = 145 Hz (2,5 ms) 3 = 0x03 = 68 Hz (5 ms) 4 = 0x04 = 34 Hz (10 ms) (voreingestellt) 5 = 0x05 = 17 Hz (21 ms) 6 = 0x06 = 8 Hz (42 ms) 7 = 0x07 = 4 Hz (84 ms) 8 = 0x08 = 2 Hz (169 ms) größer = → voreingestellter Wert Wenn IN_SAFETY = TRUE ⇨ nur der voreingestellte Wert ist zulässig. Eine andere Einstellung erzeugt einen schweren Fehler.
Qxx xx = 00...15	BOOL	Status am Binärausgang xx: Voraussetzung: Ausgang ist als Binärausgang konfiguriert TRUE: Ausgang aktiviert FALSE: Ausgang deaktiviert (= Initialwert) oder: nicht als Binärausgang konfiguriert
Qxx_FILTER xx = 00...15	BYTE	Ausgang xx: Grenzfrequenz des Software-Tiefpass-Filters erster Ordnung für die Strommessung 0 = 0x00 = kein Filter 1 = 0x01 = 390 Hz (1 ms) 2 = 0x02 = 145 Hz (2,5 ms) 3 = 0x03 = 68 Hz (5 ms) 4 = 0x04 = 34 Hz (10 ms) (voreingestellt) 5 = 0x05 = 17 Hz (21 ms) 6 = 0x06 = 8 Hz (42 ms) 7 = 0x07 = 4 Hz (84 ms) 8 = 0x08 = 2 Hz (169 ms) größer = → voreingestellter Wert

8.1.10 Systemmerker: 16 Eingänge und 32 Ausgänge (Extended-Seite)

13111

Systemmerker (Symbolname)	Typ	Beschreibung
ANALOGxx_E xx = 00...15	WORD	Extended-Analog-Eingang xx: gefilterter A/D-Wandler-Rohwert (12 Bit) ohne Kalibrierung und Normierung
CURRENTxx_E xx = 00...15	WORD	Extended-Analog-Ausgang xx: Gefilterte A/D-Wandler-Rohwerte (12 Bit) der Strommessung ohne Kalibrierung und Normierung
Ixx_E xx = 00...15	BOOL	Status am Extended-Binäreingang xx Voraussetzung: Eingang ist als Binäreingang konfiguriert (MODE = IN_DIGITAL_H oder IN_DIGITAL_L) TRUE: Spannung am Binäreingang > 70 % von VBBS FALSE: Spannung am Binäreingang < 30 % von VBBS oder: nicht als Binäreingang konfiguriert oder: falsch konfiguriert
Ixx_DFILTER_E xx = 00...11	DWORD	Extended-Impulseingang xx: Impulsdauer in [µs], die als Glitch ignoriert werden soll. Die Erfassung des Eingangssignals verzögert sich um die eingestellte Zeit. zugelassen = 0...100 000 µs voreingestellt = 0 µs = kein Filter
Ixx_FILTER_E xx = 00...15	BYTE:=4	Extended-Binär- und Analogeingang xx: Grenzfrequenz (oder Signalanstiegszeit) des Software-Tiefpass-Filters erster Ordnung 0 = 0x00 = kein Filter 1 = 0x01 = 390 Hz (1 ms) 2 = 0x02 = 145 Hz (2,5 ms) 3 = 0x03 = 68 Hz (5 ms) 4 = 0x04 = 34 Hz (10 ms) (voreingestellt) 5 = 0x05 = 17 Hz (21 ms) 6 = 0x06 = 8 Hz (42 ms) 7 = 0x07 = 4 Hz (84 ms) 8 = 0x08 = 2 Hz (169 ms) größer = → voreingestellter Wert
Qxx_E xx = 00...31	BOOL	Status am Extended-Binärausgang xx: Voraussetzung: Ausgang ist als Binärausgang konfiguriert TRUE: Ausgang aktiviert FALSE: Ausgang deaktiviert (= Initialwert) oder: nicht als Binärausgang konfiguriert
Qxx_FILTER_E xx = 00...15	BYTE	Extended-Ausgang xx: Grenzfrequenz des Software-Tiefpass-Filters erster Ordnung für die Strommessung 0 = 0x00 = kein Filter 1 = 0x01 = 390 Hz (1 ms) 2 = 0x02 = 145 Hz (2,5 ms) 3 = 0x03 = 68 Hz (5 ms) 4 = 0x04 = 34 Hz (10 ms) (voreingestellt) 5 = 0x05 = 17 Hz (21 ms) 6 = 0x06 = 8 Hz (42 ms) 7 = 0x07 = 4 Hz (84 ms) 8 = 0x08 = 2 Hz (169 ms) größer = → voreingestellter Wert

8.2 Adressbelegung und E/A-Betriebsarten

Inhalt

1656

→ auch Datenblatt

8.2.1 Adressbelegung Ein-/Ausgänge

Inhalt

2371

Eingänge: Adressbelegung (Standard-Seite) (16 Eingänge)

12787

Abkürzungen → Kapitel ***Hinweise zur Anschlussbelegung*** (→ Seite 146)
Betriebsarten der Ein- und Ausgänge → Kapitel ***Mögliche Betriebsarten Ein-/Ausgänge*** (→ Seite 436)

IEC-Adresse	Symbolische Adresse
%IX0.0 %IW2	I00 ANALOG00
%IX0.1 %IW3	I01 ANALOG01
%IX0.2 %IW4	I02 ANALOG02
%IX0.3 %IW5	I03 ANALOG03
%IX0.4 %IW6	I04 ANALOG04
%IX0.5 %IW7	I05 ANALOG05
%IX0.6 %IW8	I06 ANALOG06
%IX0.7 %IW9	I07 ANALOG07
%IX0.8 %IW10	I08 ANALOG08
%IX0.9 %IW11	I09 ANALOG09
%IX0.10 %IW12	I10 ANALOG10
%IX0.11 %IW13	I11 ANALOG11
%IX0.12 %IW14	I12 ANALOG12
%IX0.13 %IW15	I13 ANALOG13
%IX0.14 %IW16	I14 ANALOG14
%IX0.15 %IW17	I15 ANALOG15

Eingänge: Adressbelegung (Extended-Seite) (16 Eingänge)

13343

IEC-Adresse	Symbolische Adresse
%IX128.0 %IW2130	I00_E ANALOG00_E
%IX128.1 %IW131	I01_E ANALOG01_E
%IX128.2 %IW132	I02_E ANALOG02_E
%IX128.3 %IW133	I03_E ANALOG03_E
%IX128.4 %IW134	I04_E ANALOG04_E
%IX128.5 %IW135	I05_E ANALOG05_E
%IX128.6 %IW136	I06_E ANALOG06_E
%IX128.7 %IW137	I07_E ANALOG07_E
%IX128.8 %IW138	I08_E ANALOG08_E
%IX128.9 %IW139	I09_E ANALOG09_E
%IX128.10 %IW140	I10_E ANALOG10_E
%IX128.11 %IW141	I11_E ANALOG11_E
%IX128.12 %IW142	I12_E ANALOG12_E
%IX128.13 %IW143	I13_E ANALOG13_E
%IX128.14 %IW144	I14_E ANALOG14_E
%IX128.15 %IW145	I15_E ANALOG15_E

Ausgänge: Adressbelegung (Standard-Seite) (16 Ausgänge)

12940

Abkürzungen → Kapitel *Hinweise zur Anschlussbelegung* (→ Seite 146)
Betriebsarten der Ein- und Ausgänge → Kapitel *Mögliche Betriebsarten Ein-/Ausgänge* (→ Seite 436)

IEC-Adresse	Symbolische Adresse
%QX0.0 %IW18	Q00 CURRENT00
%QX0.1 %IW19	Q01 CURRENT01
%QX0.2 %IW20	Q02 CURRENT02
%QX0.3 %IW21	Q03 CURRENT03
%QX0.4 %IW22	Q04 CURRENT04
%QX0.5 %IW23	Q05 CURRENT05
%QX0.6 %IW24	Q06 CURRENT06
%QX0.7 %IW25	Q07 CURRENT07
%QX0.8 %IW26	Q08 CURRENT08
%QX0.9 %IW27	Q09 CURRENT09
%QX0.10 %IW28	Q10 CURRENT10
%QX0.11 %IW29	Q11 CURRENT11
%QX0.12 %IW30	Q12 CURRENT12
%QX0.13 %IW31	Q13 CURRENT13
%QX0.14 %IW32	Q14 CURRENT14
%QX0.15 %IW33	Q15 CURRENT15

Ausgänge: Adressbelegung (Extended-Seite) (32 Ausgänge)

13349

IEC-Adresse	Symbolische Adresse
%QX128.0 %IW18	Q00_E CURRENT00_E
%QX128.1 %IW19	Q01_E CURRENT01_E
%QX128.2 %IW20	Q02_E CURRENT02_E
%QX128.3 %IW21	Q03_E CURRENT03_E
%QX128.4 %IW22	Q04_E CURRENT04_E
%QX128.5 %IW23	Q05_E CURRENT05_E
%QX128.6 %IW24	Q06_E CURRENT06_E
%QX128.7 %IW25	Q07_E CURRENT07_E
%QX128.8 %IW26	Q08_E CURRENT08_E
%QX128.9 %IW27	Q09_E CURRENT09_E
%QX128.10 %IW28	Q10_E CURRENT10_E
%QX128.11 %IW29	Q11_E CURRENT11_E
%QX128.12 %IW30	Q12_E CURRENT12_E
%QX128.13 %IW31	Q13_E CURRENT13_E
%QX128.14 %IW32	Q14_E CURRENT14_E
%QX128.15 %IW33	Q15_E CURRENT15_E
%QX128.16	Q16_E
%QX128.17	Q17_E
%QX128.18	Q18_E
%QX128.19	Q19_E
%QX128.20	Q20_E
%QX128.21	Q21_E
%QX128.22	Q22_E
%QX128.23	Q23_E
%QX128.24	Q24_E
%QX128.25	Q25_E
%QX128.26	Q26_E
%QX128.27	Q27_E

IEC-Adresse	Symbolische Adresse
%QX128.28	Q28_E
%QX128.29	Q29_E
%QX128.30	Q30_E
%QX128.31	Q31_E

8.2.2 Mögliche Betriebsarten Ein-/Ausgänge

Inhalt

2386

Eingänge: Betriebsarten (Standard-Seite) (16 Eingänge)

14035

Mögliche Konfigurations-Kombinationen (wo zulässig) entstehen durch Addition der Konfigurations-Werte.

= diese Konfiguration ist voreingestellt

Eingänge	mögliche Betriebsart		einstellen mit FB	FB-Eingang	Wert	
					dez	hex
I00…I15	IN_NOMODE	Aus	SET_INPUT_MODE	MODE	0	0000
	IN_DIGITAL_H	plus	SET_INPUT_MODE	MODE	1	0001
	IN_DIGITAL_L	minus	SET_INPUT_MODE	MODE	2	0002
	IN_CURRENT	0…20 000 µA	SET_INPUT_MODE	MODE	4	0004
	IN_VOLTAGE10	0…10 000 mV	SET_INPUT_MODE	MODE	8	0008
	IN_VOLTAGE30	0…30 000 mV	SET_INPUT_MODE	MODE	16	0010
	IN_RATIO	0…1 000 ‰	SET_INPUT_MODE	MODE	32	0020
	Diagnose	bei IN_DIGITAL_H	SET_INPUT_MODE	DIAGNOSTICS	TRUE	
	Sicherheitskanal		SET_INPUT_MODE	SAFETY	TRUE	

Betriebsarten mit folgendem Funktionsbaustein einstellen:

FAST_COUNT (→ Seite 298)	Zählerbaustein für schnelle Eingangsimpulse
FREQUENCY (→ Seite 300)	misst die Frequenz des am gewählten Kanal ankommenden Signals
FREQUENCY_PERIOD (→ Seite 302)	misst die Frequenz und die Periodendauer (Zykluszeit) in [µs] am angegebenen Kanal
INC_ENCODER (→ Seite 304)	Vorwärts-/Rückwärts-Zählerfunktion zur Auswertung von Drehgebern
INPUT_ANALOG (→ Seite 261)	Strom- und Spannungsmessung am analogen Eingangskanal
PERIOD (→ Seite 306)	misst am angegebenen Kanal die Frequenz und die Periodendauer (Zykluszeit) in [µs]
PERIOD_RATIO (→ Seite 308)	misst die Frequenz und die Periodendauer (Zykluszeit) in [µs] über die angegebenen Perioden am angegebenen Kanal. Zusätzlich wird das Puls-/Periodenverhältnis in [‰] angegeben.
SET_INPUT_MODE (→ Seite 264)	weist einem Eingangskanal eine Betriebsart zu

Eingänge: Betriebsarten (Extended-Seite) (16 Eingänge)

13356

Mögliche Konfigurations-Kombinationen (wo zulässig) entstehen durch Addition der Konfigurations-Werte.

= diese Konfiguration ist voreingestellt

Eingänge	mögliche Betriebsart		einstellen mit FB	FB-Eingang	Wert	
					dez	hex
I00_E...I15_E	IN_NOMODE	Aus	SET_INPUT_MODE_E	MODE	0	0000
	IN_DIGITAL_H	plus	SET_INPUT_MODE_E	MODE	1	0001
	IN_DIGITAL_L	minus	SET_INPUT_MODE_E	MODE	2	0002
	IN_CURRENT	0…20 000 µA	SET_INPUT_MODE_E	MODE	4	0004
	IN_VOLTAGE10	0…10 000 mV	SET_INPUT_MODE_E	MODE	8	0008
	IN_VOLTAGE30	0…30 000 mV	SET_INPUT_MODE_E	MODE	16	0010
	IN_RATIO	0…1 000 ‰	SET_INPUT_MODE_E	MODE	32	0020
	Diagnose	bei IN_DIGITAL_H	SET_INPUT_MODE_E	DIAGNOSTICS	TRUE	

Betriebsarten mit folgendem Funktionsbaustein einstellen:

FAST_COUNT_E	= FAST_COUNT für die Extended-Seite
FREQUENCY_E	= FREQUENCY für die Extended-Seite
FREQUENCY_PERIOD_E	= FREQUENCY_PERIOD für die Extended-Seite
INC_ENCODER_E	= INC_ENCODER für die Extended-Seite
INPUT_ANALOG_E	= INPUT_ANALOG für die Extended-Seite
PERIOD_E	= PERIOD für die Extended-Seite
PERIOD_RATIO_E	= PERIOD_RATIO für die Extended-Seite
SET_INPUT_MODE_E	= SET_INPUT_MODE für die Extended-Seite

Ausgänge: Betriebsarten (Standard-Seite) (16 Ausgänge)

14036

= diese Konfiguration ist voreingestellt

Ausgänge	mögliche Betriebsart		einstellen mit FB	FB-Eingang	Wert	
					dez	hex
Q00...Q15	OUT_DIGITAL_H	plus	SET_OUTPUT_MODE	MODE	1	0001
	OUT_DIGITAL_L	minus	SET_OUTPUT_MODE	MODE	2	0002
	Diagnose	bei OUT_DIGITAL_H	SET_OUTPUT_MODE	DIAGNOSTICS	TRUE	
	Überlastschutz	bei OUT_DIGITAL_H mit Strommessung	SET_OUTPUT_MODE	PROTECTION	TRUE	
	Strommessbereich	keine Strommessung	SET_OUTPUT_MODE	CURRENT_RANGE	0	00
	Strommessbereich	2 A	SET_OUTPUT_MODE	CURRENT_RANGE	1	01
	Strommessbereich	4 A	SET_OUTPUT_MODE	CURRENT_RANGE	2	02
	Sicherheitskanal	bei OUT_DIGITAL_H	SET_OUTPUT_MODE	SAFETY	TRUE	

Betriebsarten mit folgendem Funktionsbaustein einstellen:

OUTPUT_BRIDGE (→ Seite 326)	H-Brücke an einem PWM-Kanalpaar
OUTPUT_CURRENT (→ Seite 330)	misst den Strom (Mittelung über Dither-Periode) an einem Ausgangskanal
OUTPUT_CURRENT_CONTROL (→ Seite 331)	Stromregler für einen PWMi-Ausgangskanal
PWM1000 (→ Seite 333)	initialisiert und parametriert einen PWM-fähigen Ausgangskanal das Puls-Pausen-Verhältnis kann in 1 ‰-Schritten angegeben werden
SET_OUTPUT_MODE (→ Seite 313)	setzt die Betriebsart des gewählten Ausgangskanals

Ausgänge: zulässige Betriebsarten

15547

Betriebsart		Q00	Q01	Q02	Q03	Q04	Q05	Q06	Q07
OUT_NOMODE	Aus	X	X	X	X	X	X	X	X
OUT_DIGITAL_H	plus	X	X	X	X	X	X	X	X
OUT_DIGITAL_L	minus	--	X	--	X	--	--	--	--
Diagnose	bei OUT_DIGITAL_H	X	X	X	X	X	X	X	X
Überlastschutz	bei OUT_DIGITAL_H mit Strommessung	X	X	X	X	X	X	X	X
Strommessbereich	2 A	X	X	X	X	--	--	--	--
Strommessbereich	3 A	--	--	--	--	X	X	X	X
Strommessbereich	4 A	X	X	X	X	--	--	--	--
Sicherheitskanal		X	X	X	X	X	X	X	X
PWM		X	X	X	X	X	X	X	X
H-Brücke		--	X	--	X	--	--	--	--

Betriebsart		Q08	Q09	Q10	Q11	Q12	Q13	Q14	Q15
OUT_NOMODE	Aus	X	X	X	X	X	X	X	X
OUT_DIGITAL_H	plus	X	X	X	X	X	X	X	X
OUT_DIGITAL_L	minus	--	X	--	X	--	--	--	--
Diagnose	bei OUT_DIGITAL_H	X	X	X	X	X	X	X	X
Überlastschutz	bei OUT_DIGITAL_H mit Strommessung	X	X	X	X	X	X	X	X
Strommessbereich	2 A	X	X	X	X	--	--	--	--
Strommessbereich	3 A	--	--	--	--	X	X	X	X
Strommessbereich	4 A	X	X	X	X	--	--	--	--
Sicherheitskanal		X	X	X	X	X	X	X	X
PWM		X	X	X	X	X	X	X	X
H-Brücke		--	X	--	X	--	--	--	--

Ausgänge: Betriebsarten (Extended-Seite) (32 Ausgänge)

13357

= diese Konfiguration ist voreingestellt

Ausgänge	mögliche Betriebsart		einstellen mit FB	FB-Eingang	Wert	
					dez	hex
Q00_E ...Q15_E	OUT_DIGITAL_H	plus	SET_OUTPUT_MODE	MODE	1	0001
	OUT_DIGITAL_L	minus	SET_OUTPUT_MODE	MODE	2	0002
	Diagnose	bei OUT_DIGITAL_H	SET_OUTPUT_MODE	DIAGNOSTICS	TRUE	
	Überlastschutz	bei OUT_DIGITAL_H mit Strommessung	SET_OUTPUT_MODE	PROTECTION	TRUE	
	Strommessbereich	keine Strommessung	SET_OUTPUT_MODE	CURRENT_RANGE	0	00
	Strommessbereich	2 A / 3 A	SET_OUTPUT_MODE	CURRENT_RANGE	1	01
	Strommessbereich	4 A	SET_OUTPUT_MODE	CURRENT_RANGE	2	02
Q16_E ...Q31_E	OUT_DIGITAL_H	plus	SET_OUTPUT_MODE	MODE	1	0001
	Diagnose	bei OUT_DIGITAL_H	SET_OUTPUT_MODE	DIAGNOSTICS	FALSE	
	Strommessbereich	keine Strommessung	SET_OUTPUT_MODE	CURRENT_RANGE	0	00

Betriebsarten mit folgendem Funktionsbaustein einstellen:

OUTPUT_BRIDGE_E	= OUTPUT_BRIDGE für die Extended-Seite
OUTPUT_CURRENT_E	= OUTPUT_CURRENT für die Extended-Seite
OUTPUT_CURRENT_CONTROL_E	= OUTPUT_CURRENT_CONTROL für die Extended-Seite
PWM1000_E	= PWM1000 für die Extended-Seite
SET_OUTPUT_MODE_E	= SET_OUTPUT_MODE für die Extended-Seite

Ausgänge: zulässige Betriebsarten

15557

Betriebsart		Q00_E	Q01_E	Q02_E	Q03_E	Q04_E	Q05_E	Q06_E	Q07_E
OUT_NOMODE	Aus	X	X	X	X	X	X	X	X
OUT_DIGITAL_H	plus	X	X	X	X	X	X	X	X
OUT_DIGITAL_L	minus	--	X	--	X	--	--	--	--
Diagnose	bei OUT_DIGITAL_H	X	X	X	X	X	X	X	X
Überlastschutz	bei OUT_DIGITAL_H mit Strommessung	X	X	X	X	X	X	X	X
Strommessbereich	2 A	X	X	X	X	X	X	X	X
Strommessbereich	4 A	X	X	X	X	--	--	--	--
PWM		X	X	X	X	X	X	X	X
H-Brücke		--	X	--	X	--	--	--	--

Betriebsart		Q08_E	Q09_E	Q10_E	Q11_E	Q12_E	Q13_E	Q14_E	Q15_E
OUT_NOMODE	Aus	X	X	X	X	X	X	X	X
OUT_DIGITAL_H	plus	X	X	X	X	X	X	X	X
OUT_DIGITAL_L	minus	--	X	--	X	--	--	--	--
Diagnose	bei OUT_DIGITAL_H	X	X	X	X	X	X	X	X
Überlastschutz	bei OUT_DIGITAL_H mit Strommessung	X	X	X	X	X	X	X	X
Strommessbereich	2 A	X	X	X	X	X	X	X	X
Strommessbereich	4 A	X	X	X	X	--	--	--	--
PWM		X	X	X	X	X	X	X	X
H-Brücke		--	X	--	X	--	--	--	--

Betriebsart		Q16_E	Q17_E	Q18_E	Q19_E	Q20_E	Q21_E	Q22_E	Q23_E
OUT_NOMODE	Aus	X	X	X	X	X	X	X	X
OUT_DIGITAL_H	plus	X	X	X	X	X	X	X	X

Betriebsart		Q24_E	Q25_E	Q26_E	Q27_E	Q28_E	Q29_E	Q30_E	Q31_E
OUT_NOMODE	Aus	X	X	X	X	X	X	X	X
OUT_DIGITAL_H	plus	X	X	X	X	X	X	X	X

8.2.3 Adressen / Variablen der E/As

Inhalt

2376

Eingänge: Adressen und Variablen (Standard-Seite) (16 Eingänge)

13358

IEC-Adresse	E/A-Variable	Bemerkung
%IB0	I0	Eingangsbyte 0 (%IX0.0...%IX0.7)
%QB36	I00_FILTER	Filterbyte für %IX0.0 / %IW2
%QB37	I01_FILTER	Filterbyte für %IX0.1 / %IW3
%QB38	I02_FILTER	Filterbyte für %IX0.2 / %IW4
%QB39	I03_FILTER	Filterbyte für %IX0.3 / %IW5
%QB40	I04_FILTER	Filterbyte für %IX0.4 / %IW6
%QB41	I05_FILTER	Filterbyte für %IX0.5 / %IW7
%QB42	I06_FILTER	Filterbyte für %IX0.6 / %IW8
%QB43	I07_FILTER	Filterbyte für %IX0.7 / %IW9
%QD17	I00_DFILTER	Filterwert Zähl-/Impulseingang 0
%QD18	I01_DFILTER	Filterwert Zähl-/Impulseingang 1
%QD19	I02_DFILTER	Filterwert Zähl-/Impulseingang 2
%QD20	I03_DFILTER	Filterwert Zähl-/Impulseingang 3
%QD21	I04_DFILTER	Filterwert Zähl-/Impulseingang 4
%QD22	I05_DFILTER	Filterwert Zähl-/Impulseingang 5
%QD23	I06_DFILTER	Filterwert Zähl-/Impulseingang 6
%QD24	I07_DFILTER	Filterwert Zähl-/Impulseingang 7
%QD25	I08_DFILTER	Filterwert Zähl-/Impulseingang 8
%QD26	I09_DFILTER	Filterwert Zähl-/Impulseingang 9
%QD27	I10_DFILTER	Filterwert Zähl-/Impulseingang 10
%QD28	I11_DFILTER	Filterwert Zähl-/Impulseingang 11
%IB1	I1	Eingangsbyte 1 (%IX0.8...%IX0.15)
%QB44	I08_FILTER	Filterbyte für %IX0.8 / %IW10
%QB45	I09_FILTER	Filterbyte für %IX0.9 / %IW11
%QB46	I10_FILTER	Filterbyte für %IX0.10 / %IW12
%QB47	I11_FILTER	Filterbyte für %IX0.11/ %IW13
%QB48	I12_FILTER	Filterbyte für %IX0.12 / %IW14
%QB49	I13_FILTER	Filterbyte für %IX0.13 / %IW15
%QB50	I14_FILTER	Filterbyte für %IX0.14 / %IW16
%QB51	I15_FILTER	Filterbyte für %IX0.15 / %IW17
%IW0		Eingangswort (%IX0.00...%IX0.15)
%MW3976	ERROR_CURRENT_I0	Fehlerwort Überstrom (%MX3976.0...%MX3976.15)
%MW3977	ERROR_SHORT_I0	Fehlerwort Kurzschluss (%MX3977.0...%MX3977.15)
%MW3978	ERROR_BREAK_I0	Fehlerwort Leiterbruch (%MX3978.0...%MX3978.15)

Eingänge: Adressen und Variablen (Extended-Seite) (16 Eingänge)

13359

IEC-Adresse	E/A-Variable	Bemerkung
%IB128	I0_E	Eingangsbyte 0 (%IX128.0...%IX128.7)
%QB308	I00_FILTER_E	Filterbyte für %IX128.0 / %IW2
%QB309	I01_FILTER_E	Filterbyte für %IX128.1 / %IW3
%QB310	I02_FILTER_E	Filterbyte für %IX128.2 / %IW4
%QB311	I03_FILTER_E	Filterbyte für %IX128.3 / %IW5
%QB312	I04_FILTER_E	Filterbyte für %IX128.4 / %IW6
%QB313	I05_FILTER_E	Filterbyte für %IX128.5 / %IW7
%QB314	I06_FILTER_E	Filterbyte für %IX128.6 / %IW8
%QB315	I07_FILTER_E	Filterbyte für %IX128.7 / %IW9
%QD85	I00_DFILTER_E	Filterwert Zähl-/Impulseingang 0
%QD86	I01_DFILTER_E	Filterwert Zähl-/Impulseingang 1
%QD87	I02_DFILTER_E	Filterwert Zähl-/Impulseingang 2
%QD88	I03_DFILTER_E	Filterwert Zähl-/Impulseingang 3
%QD89	I04_DFILTER_E	Filterwert Zähl-/Impulseingang 4
%QD90	I05_DFILTER_E	Filterwert Zähl-/Impulseingang 5
%QD91	I06_DFILTER_E	Filterwert Zähl-/Impulseingang 6
%QD92	I07_DFILTER_E	Filterwert Zähl-/Impulseingang 7
%QD93	I08_DFILTER_E	Filterwert Zähl-/Impulseingang 8
%QD94	I09_DFILTER_E	Filterwert Zähl-/Impulseingang 9
%QD95	I10_DFILTER_E	Filterwert Zähl-/Impulseingang 10
%QD96	I11_DFILTER_E	Filterwert Zähl-/Impulseingang 11
%IB129	I1_E	Eingangsbyte 1 (%IX128.8...%IX128.15)
%QB316	I08_FILTER_E	Filterbyte für %IX128.8 / %IW10
%QB317	I09_FILTER_E	Filterbyte für %IX128.9 / %IW11
%QB318	I10_FILTER_E	Filterbyte für %IX128.10 / %IW12
%QB319	I11_FILTER_E	Filterbyte für %IX128.11/ %IW13
%QB320	I12_FILTER_E	Filterbyte für %IX128.12 / %IW14
%QB321	I13_FILTER_E	Filterbyte für %IX128.13 / %IW15
%QB322	I14_FILTER_E	Filterbyte für %IX128.14 / %IW16
%QB323	I15_FILTER_E	Filterbyte für %IX128.15 / %IW17

Ausgänge: Adressen und Variablen (Standard-Seite) (16 Ausgänge)

13360

IEC-Adresse	E/A-Variable	Bemerkung
%QB0		Ausgangsbyte 0 (%QX0.0...%QX0.7)
%QB52	Q00_FILTER	Filter-Byte für %IW18
%QB53	Q01_FILTER	Filter-Byte für %IW19
%QB54	Q02_FILTER	Filter-Byte für %IW20
%QB55	Q03_FILTER	Filter-Byte für %IW21
%QB56	Q04_FILTER	Filter-Byte für %IW22
%QB57	Q05_FILTER	Filter-Byte für %IW23
%QB58	Q06_FILTER	Filter-Byte für %IW24
%QB59	Q07_FILTER	Filter-Byte für %IW25
%IW18	CURRENT00	Ausgangsstrom (Rohwert) an Q00
%IW19	CURRENT01	Ausgangsstrom (Rohwert) an Q01
%IW20	CURRENT02	Ausgangsstrom (Rohwert) an Q02
%IW21	CURRENT03	Ausgangsstrom (Rohwert) an Q03
%IW22	CURRENT04	Ausgangsstrom (Rohwert) an Q04
%IW23	CURRENT05	Ausgangsstrom (Rohwert) an Q05
%IW24	CURRENT06	Ausgangsstrom (Rohwert) an Q06
%IW25	CURRENT07	Ausgangsstrom (Rohwert) an Q07
%QB1		Ausgangsbyte 1 (%QX0.8...%QX0.15)
%QB60	Q08_FILTER	Filter-Byte für %IW26
%QB61	Q09_FILTER	Filter-Byte für %IW27
%QB62	Q10_FILTER	Filter-Byte für %IW28
%QB63	Q11_FILTER	Filter-Byte für %IW29
%QB64	Q12_FILTER	Filter-Byte für %IW30
%QB65	Q13_FILTER	Filter-Byte für %IW31
%QB66	Q14_FILTER	Filter-Byte für %IW32
%QB67	Q15_FILTER	Filter-Byte für %IW33
%IW26	CURRENT08	Ausgangsstrom (Rohwert) an Q08
%IW27	CURRENT09	Ausgangsstrom (Rohwert) an Q09
%IW28	CURRENT10	Ausgangsstrom (Rohwert) an Q10
%IW29	CURRENT11	Ausgangsstrom (Rohwert) an Q11
%IW30	CURRENT12	Ausgangsstrom (Rohwert) an Q12
%IW31	CURRENT13	Ausgangsstrom (Rohwert) an Q13
%IW32	CURRENT14	Ausgangsstrom (Rohwert) an Q14
%IW33	CURRENT15	Ausgangsstrom (Rohwert) an Q15
%QW0		Ausgangswort 0 (%QX0.0...%QX0.15)
%MW3973	ERROR_SHORT_Q0	Fehlerwort Kurzschluss (%QX0.0...%QX0.15)
%MW3974	ERROR_BREAK_Q0	Fehlerwort Leiterbruch (%QX0.0...%QX0.15)
%MW3975	ERROR_CONTROL_Q0	Fehlerwort Stromregelung (%QX0.0...%QX0.15)

Ausgänge: Adressen und Variablen (Extended-Seite) (32 Ausgänge)

13361

IEC-Adresse	E/A-Variable	Bemerkung
%QB2		Ausgangsbyte 2 (%QX128.0...%QX128.7)
%QB324	Q00_FILTER_E	Filter-Byte für %IW18
%QB325	Q01_FILTER_E	Filter-Byte für %IW19
%QB326	Q02_FILTER_E	Filter-Byte für %IW20
%QB327	Q03_FILTER_E	Filter-Byte für %IW21
%QB328	Q04_FILTER_E	Filter-Byte für %IW22
%QB329	Q05_FILTER_E	Filter-Byte für %IW23
%QB330	Q06_FILTER_E	Filter-Byte für %IW24
%QB331	Q07_FILTER_E	Filter-Byte für %IW25
%IW146	CURRENT00_E	Ausgangsstrom (Rohwert) an Q00_E
%IW147	CURRENT01_E	Ausgangsstrom (Rohwert) an Q01_E
%IW148	CURRENT02_E	Ausgangsstrom (Rohwert) an Q02_E
%IW149	CURRENT03_E	Ausgangsstrom (Rohwert) an Q03_E
%IW150	CURRENT04_E	Ausgangsstrom (Rohwert) an Q04_E
%IW151	CURRENT05_E	Ausgangsstrom (Rohwert) an Q05_E
%IW152	CURRENT06_E	Ausgangsstrom (Rohwert) an Q06_E
%IW153	CURRENT07_E	Ausgangsstrom (Rohwert) an Q07_E
%QB3		Ausgangsbyte 3 (%QX128.8...%QX128.15)
%QB332	Q08_FILTER_E	Filter-Byte für %IW26
%QB333	Q09_FILTER_E	Filter-Byte für %IW27
%QB334	Q10_FILTER_E	Filter-Byte für %IW28
%QB335	Q11_FILTER_E	Filter-Byte für %IW29
%QB336	Q12_FILTER_E	Filter-Byte für %IW30
%QB337	Q13_FILTER_E	Filter-Byte für %IW31
%QB338	Q14_FILTER_E	Filter-Byte für %IW32
%QB339	Q15_FILTER_E	Filter-Byte für %IW33
%IW154	CURRENT08_E	Ausgangsstrom (Rohwert) an Q08_E
%IW155	CURRENT09_E	Ausgangsstrom (Rohwert) an Q09_E
%IW156	CURRENT10_E	Ausgangsstrom (Rohwert) an Q10_E
%IW157	CURRENT11_E	Ausgangsstrom (Rohwert) an Q11_E
%IW158	CURRENT12_E	Ausgangsstrom (Rohwert) an Q12_E
%IW159	CURRENT13_E	Ausgangsstrom (Rohwert) an Q13_E
%IW160	CURRENT14_E	Ausgangsstrom (Rohwert) an Q14_E
%IW161	CURRENT15_E	Ausgangsstrom (Rohwert) an Q15_E
%QB4		Ausgangsbyte 4 (%QX128.16...%QX128.23)
%QB5		Ausgangsbyte 5 (%QX128.24...%QX128.31)

8.3 CANopen-Tabellen

Inhalt

9941

Die folgenden Tabellen informieren Sie über wichtige Werte und Einstellungen der CANopen-Schnittstellen.

8.3.1 Aufbau von CANopen-Meldungen

Inhalt

9971

Eine CANopen-Meldung besteht aus der COB-ID und bis zu 8 Bytes Daten:

COB-ID			DLC	Byte 1		Byte 2		Byte 3		Byte 4		Byte 5		Byte 6		Byte 7		Byte 8	
X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X

Details erfahren Sie in den folgenden Kapiteln.

Beachten Sie die umgekehrte Byte-Reihenfolge! (⇨ Little Endian oder Intel-Format)

Beispiele:

Wert [hex]	Datentyp	Byte 1		Byte 2		Byte 3		Byte 4		Byte 5		Byte 6		Byte 7		Byte 8	
12	BYTE	1	2	–	–	–	–	–	–	–	–	–	–	–	–	–	–
1234	WORD	3	4	1	2	–	–	–	–	–	–	–	–	–	–	–	–
12345678	DWORD	7	8	5	6	3	4	1	2	–	–	–	–	–	–	–	–

Aufbau der COB-ID

9972

Der erste Teil einer Meldung ist die COB-ID. Aufbau der 11-Bit COB-ID:

Nibble 0				Nibble 1				Nibble 2			
11	10	9	8	7	6	5	4	3	2	1	0
--	**3**	**2**	**1**	**0**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
--	**Funktions-Code**				**Node-ID**						

Die COB-ID besteht aus ***Funktions-Code / Predefined Connectionset*** (→ Seite 450) und Node-ID.

Beispiel:
Das Kommunikations-Objekt = TPDO1 (TX)
Die Knoten-Nummer des Geräts = 0x020 = 32

Berechnung:
Der Funktions-Code für das Kommunikations-Objekt TPDO1 = 0x03
Die Wertigkeit des Funktions-Code in der 11-Bit-COB-ID = 0x03 • 0x80 = 0x180
Dazu die Knoten-Nummer (0x020) addieren ⇨ die COB-ID lautet: 0x1A0

1				A				0			
3	2	1	0	3	2	1	0	3	2	1	0
0	**0**	**0**	**1**	**1**	**0**	**1**	**0**	**0**	**0**	**0**	**0**
--	0x03 = 3				0x020 = 32						

Funktions-Code / Predefined Connectionset

9966

Im "CANopen Predefined Connectionset" sind einige Funktions-Codes vorbelegt.

Wenn Sie das Predefined Connectionset verwenden, können Sie ein CANopen-Netzwerk von bis zu 127 Teilnehmern in Betrieb nehmen, ohne dass es zu einer doppelten Vergabe von COB-IDs kommt.

Broadcast- oder Multicast-Nachrichten:

Kommunikations-Objekt	Funktions-Code [hex]	COB-ID [hex]	zugehörige Parameter-Objekte [hex]
NMT	0	000	
SYNC	1	080	1005, 1006, 1007, 1028
TIME	2	100	1012, 1013

Punkt-zu-Punkt-Nachrichten:

Kommunikations-Objekt	Funktions-Code [hex]	COB-ID [hex]	zugehörige Parameter-Objekte [hex]
EMERGENCY	1	080 + Node-ID	1014, 1015
TPDO1 (TX)	3	180 + Node-ID	1800
RPDO1 (RX)	4	200 + Node-ID	1400
TPDO2 (TX)	5	280 + Node-ID	1801
RPDO2 (RX)	6	300 + Node-ID	1401
TPDO3 (TX)	7	380 + Node-ID	1802
RPDO3 (RX)	8	400 + Node-ID	1402
TPDO4 (TX)	9	480 + Node-ID	1803
RPDO4 (RX)	A	500 + Node-ID	1403
Default SSDO (TX)	B	580 + Node-ID	1200
Default CSDO (RX)	C	600 + Node-ID	1280
NMT Error Control	E	700 + Node-ID	1016, 1017

TX = Slave sendet an Master
RX = Slave empfängt von Master

SSDO = Server-SDO
CSDO = Client-SDO

SDO-Kommando-Bytes

9968

Aufbau einer SDO-Nachricht:

COB-ID	DLC	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7	Byte 8
XXX	8	Kommando	Index		Sub-Index	Daten abhängig von den zu transportierenden Daten			

Beachten Sie die umgekehrte Byte-Reihenfolge! (⇨ Little Endian oder Intel-Format)

Eine SDO-COB-ID setzt sich wie folgt zusammen:

CANopen	
Node-ID	**COB-ID SDO**
1…127	TX: 0x580 + Node-ID
	RX: 0x600 + Node-ID

TX = Slave sendet an Master
RX = Slave empfängt von Master

DLC = **D**ata **L**ength **C**ode = bei CANopen die Anzahl der Daten-Bytes in einer Nachricht.
Für →SDO: DLC = 8

SDO-Kommando-Bytes:

Kommando hex	dez	Nachricht	Datenlänge	Beschreibung
21	33	Anforderung	mehr als 4 Bytes	Daten an Slave senden
22	34	Anforderung	1…4 Bytes	Daten an Slave senden
23	35	Anforderung	4 Bytes	Daten an Slave senden
27	39	Anforderung	3 Bytes	Daten an Slave senden
2B	43	Anforderung	2 Bytes	Daten an Slave senden
2F	47	Anforderung	1 Byte	Daten an Slave senden
40	64	Anforderung	---	Daten von Slave anfordern
42	66	Antwort	1…4 Bytes	Daten von Slave an Master senden
43	67	Antwort	4 Bytes	Daten von Slave an Master senden
47	71	Antwort	3 Bytes	Daten von Slave an Master senden
4B	75	Antwort	2 Bytes	Daten von Slave an Master senden
4F	79	Antwort	1 Byte	Daten von Slave an Master senden
60	96	Antwort	---	Datentransfer in Ordnung: Empfangsbestätigung von Slave an Master senden
80	128	Antwort	4 Bytes	Datentransfer fehlgeschlagen: Abbruch-Nachricht von Slave an Master senden → Kapitel *SDO-Abbruch-Code* (→ Seite 452)

SDO-Abbruch-Code

9970

Der SDO-Abbruch-Code gehört NICHT zum Emergency-Telegramm!

Abbruch-Code [hex]	Beschreibung
0503 0000	toggle bit not alternated
0504 0000	SDO protocol timed out
0504 0001	client/server command specifier not valid or unknown
0504 0002	invalid block size (block mode only)
0504 0003	invalid sequence number (block mode only)
0504 0004	CRC error (block mode only)
0504 0005	out of memory
0601 0000	unsupported access to an object
0601 0001	attempt to read a write only object
0601 0002	attempt to write a read only object
0602 0000	object does not exist in the object dictionary
0604 0041	object cannot be mapped to the PDO
0604 0042	the number and length of the objects to be mapped would exceed PDO length
0604 0043	general parameter incompatibility reason
0604 0047	general internal incompatibility in the device
0606 0000	access failed due to an hardware error
0607 0010	data type does not match, length of service parameter does not match
0607 0012	data type does not match, length of service parameter too high
0607 0013	data type does not match, length of service parameter too low
0609 0011	sub-index does not exist
0609 0030	value range of parameter exceeded (only for write access)
0609 0031	value of parameter written too high
0609 0032	value of parameter written too low
0609 0036	maximum value is less than minimum value
0800 0000	general error
0800 0020	data cannot be transferred or stored to the application
0800 0021	data cannot be transferred or stored to the application because of local control
0800 0022	data cannot be transferred or stored to the application because of the present device state
0800 0023	object dictionary dynamic generation fails or no object dictionary is present (e.g. object dictionary is generated from file and generation fails because of an file error)

8.3.2 Bootup-Nachricht

9961

Der CAN-Teilnehmer sendet nach dem Booten einmalig die Bootup-Nachricht:

	COB-ID	DLC	Byte 1
hex	0x700 + Node-ID	0x1	0x00
dez	1 792 + Node-ID	1	0

Somit ist der Teilnehmer im CAN-Netzwerk lauffähig.

DLC = **D**ata **L**ength **C**ode = bei CANopen die Anzahl der Daten-Bytes in einer Nachricht.
Für →SDO: DLC = 8

Beispiel:
Die Node-ID des Teilnehmers ist 0x7D = 125.
Dann lautet die COB-ID der Bootup-Nachricht: 0x77D = 1 917

Abweichung:
Es gibt Geräte, die kein [0x700 + Node-ID] senden können (das sind Geräte, die vor der Version 4 der CANopen-Spezifikation entstanden sind).
Diese Geräte senden stattdessen folgende Bootup-Nachricht und ohne Status:

	COB-ID	DLC
hex	0x080 + Node-ID	0x0
dez	128 + Node-ID	0

8.3.3 Netzwerk-Management (NMT)

Inhalt

9974

Netzwerk-Management-Kommandos

9962

Mit folgenden Netzwerk-Management-Kommandos kann der Anwender den Betriebsmodus von einzelnen oder allen CAN-Teilnehmern beeinflussen. Muster:

COB-ID	DLC	Byte 1	Byte 2
0x000	X	Kommando	Node-ID

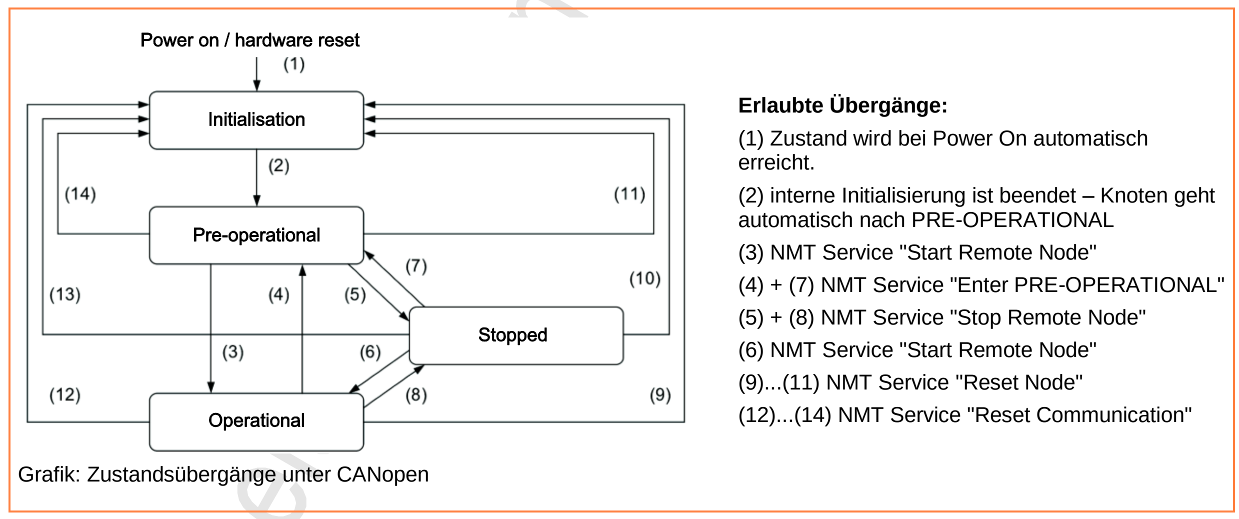
Node-ID = 00 ⇨ Kommando gilt zeitgleich für alle Knoten im Netz

COB-ID	NMT-Kommando		Beschreibung	
0x000	0x01 = 01	Node-ID	start_remode_node	Knoten in den Zustand Operational versetzen
0x000	0x02 = 02	Node-ID	stop_remode_node	Knoten in den Zustand STOPPED versetzen
0x000	0x80 = 128	Node-ID	enter_pre-operational	Knoten in den Zustand PRE-OPERATIONAL versetzen
0x000	0x81 = 129	Node-ID	reset_node	Knoten zurücksetzen
0x000	0x82 = 130	Node-ID	reset_communication	CAN-Kommunikation des Knotens zurücksetzen

NMT-Status

9963

Das Status-Byte gibt Auskunft über den Zustand des CAN-Teilnehmers.



Erlaubte Übergänge:

(1) Zustand wird bei Power On automatisch erreicht.

(2) interne Initialisierung ist beendet – Knoten geht automatisch nach PRE-OPERATIONAL

(3) NMT Service "Start Remote Node"

(4) + (7) NMT Service "Enter PRE-OPERATIONAL"

(5) + (8) NMT Service "Stop Remote Node"

(6) NMT Service "Start Remote Node"

(9)...(11) NMT Service "Reset Node"

(12)...(14) NMT Service "Reset Communication"

Grafik: Zustandsübergänge unter CANopen

NMT-Status für CANopen-Master

9964

> Diese Status zeigen den internen Zustand des CANopen-Master-Stack.
> Sie sind nicht durch die CANopen-Spezifikation vorgegeben.

Status hex	dez	Beschreibung
00	0	nicht definiert
01	1	Master wartet auf die Bootup-Nachricht des Slaves. ODER: Master wartet auf Ablauf der GuardTime.
02	2	• Master wartet 300 ms. • Master fordert das Objekt 0x1000 an. • Danach wechselt der Master auf Status 3.
03	3	Der Master konfiguriert seine Slaves. Dazu sendet der Master an die Slaves der Reihe nach alle vom Konfigurator erzeugten SDOs. Danach wechselt der Master auf Status 5.
05	5	Nachdem an alle Slaves die SDOs übertragen wurden, geht der Master in den Status 5 und bleibt in diesem Status. Status 5 ist für den Master der normale Betriebszustand.

Knoten-Status aus FB lesen:

verwendeter Funktionsbaustein	hier steht dieser Knoten-Status
CANx_MASTER_STATUS	Ausgang NODE_STATE
CANOPEN_GETSTATE	Ausgang MASTERSTATE

NMT-Status für CANopen-Slave

9965

> Diese Status zeigen den internen Zustand des CANopen-Master-Stack im Bezug auf die Initialisierung eines CANopen-Slave.
> Sie sind nicht durch die CANopen-Spezifikation vorgegeben.

Die Struktur CANx_NODE_STATE liegt in einem Array, dessen Adresse dem FB CANx_MASTER_STATUS über den Eingang NOTE_STATE_SLAVES übergeben werden muss.
Die folgenden Werte kann in der Struktur CANx_NODE_STATE der Ausgang NODE_STATE annehmen:

Status hex	dez	Beschreibung
FF	-1	Initialer Status Der CANopen-Slave wird durch das NMT-Kommando [Reset_Node] zurückgesetzt. Anschließend Wechsel in den Status 1. Ist in der CODESYS-Steuerungskonfiguration beim CANopen-Slave die Option [nicht initialisieren] aktiviert, wird der Status -1 übersprungen und der Status 1 ist der initiale Status.
00	0	nicht definiert
01	1	Warten auf die Bootup-Nachricht vom Slave. Nach dem Empfang der Bootup-Nachricht ODER spätestens nach 2 s Wartezeit Wechsel in den Status 2.
02	2	Auslesen des Objekts 0x1000 aus dem Objektverzeichnis des CANopen-Slaves per SDO-Zugriff. Nach einer Antwort vom CANopen-Slave ODER einer Wartezeit von 500 ms erfolgt Wechsel in den Status 3. Ist der CANopen-Slave in der CODESYS-Steuerungskonfiguration als "optional" konfiguriert, erfolgt nach Ablauf der Wartezeit ein Wechsel in den Status 97. Entspricht der aus dem Objekt 0x1000 ausgelesene Gerätetyp nicht der Angabe der in der CODESYS-Steuerungskonfiguration eingebundenen EDS-Datei, erfolgt zwar ein Wechsel in den Status 3, aber am Ende von Status 3 ein Wechsel in den Status 98.

Status hex	dez		Beschreibung
03	3	Der CANopen-Slave wird vom Master per SDO-Zugriff konfiguriert. Ist in der CODESYS-Steuerungskonfiguration beim CANopen-Slave die Option [Knoten zurücksetzen] aktiviert, wird während der ersten Konfiguration die Zeichenkette "load" an das Objekt 0x1011/01 gesendet und anschließend der CANopen-Slave mit dem NMT-Kommando [Reset_Node] neu gestartet. Anschließend Wechsel in den Status 1, der [load]-Befehl mit anschließendem Reset wird im weiteren Verlauf im Status 3 nicht mehr ausgeführt. CANopen-Slaves, bei denen während der Konfigurationsphase ein Problem auftritt, bleiben entweder im Status 3 oder wechseln in einen Fehlerstatus (Status > 5). Über das Strukturelement SET_TIMEOUT_STATE der Struktur CANx_NODE_STATE ist es möglich, einen nicht vorhandenen CANopen-Slave, der in der CODESYS-Steuerungskonfiguration nicht als "optional" konfiguriert wurde, in den Status 4 wechseln zu lassen. Ansonsten würde der fehlende CANopen-Slave die Initialisierung des CANopen-Netzwerks blockieren.
04	4	CANopen-Slave ist konfiguriert und im CANopen-Status PRE-OPERATIONAL. Befinden sich alle CANopen-Slaves im Zustand 4 ODER 97 und ist in der CODESYS-Steuerungskonfiguration beim CANopen-Master die Option [Automatisch starten] aktiviert, wird das NMT-Kommando [start] versendet. Ist in der CODESYS-Steuerungskonfiguration beim CANopen-Master die Option [Automatisch starten] nicht aktiviert, müssen die CANopen-Slaves manuell über das ihnen zugeordnete Strukturelement START_NODE der Struktur CANx_NODE_STATE oder alle zusammen über den Eingang START_ALL_NODES des FB CANx_MASTER_STATUS gestartet werden. Anschließend Wechsel in den Status 5.
05	5	[Normal Operation], der CANopen-Slave ist im CANopen-Status OPERATIONAL. PDOs werden übertragen.
61	97	CANopen-Slave ist als [optional] konfiguriert und ein Zugriff auf das Objekt 0x1000 blieb ohne Antwort. Wird im späteren Verlauf eine Bootup-Nachricht vom CANopen-Slave empfangen und ist in der CODESYS-Steuerungskonfiguration beim CANopen-Master die Option [Automatisch starten] aktiviert, erfolgt ein Wechsel in den Status 2.
62	98	Gerätetyp im Objekt 0x1000 entspricht nicht dem Wert in der EDS-Datei, die in der CODESYS-Steuerungskonfiguration für den CANopen-Slave eingebunden wurde. Wechsel in den Zustand 4 über das Strukturelement SET_NODE_STATE der Struktur CANx_NODE_STATE möglich. Sollte der CANopen-Slave über das globale NMT-Kommando [start] (Node-ID = 0) in den CANopen-Zustand OPERATIONAL versetzt worden sein, werden keine PDOs vom CANopen-Master an den CANopen-Slave versendet und empfangene PDOs werden ignoriert.
63	99	Es ist ein Node-Guarding oder Heartbeat Timeout aufgetreten. Sobald der CANopen-Slave wieder auf Node-Guarding reagiert bzw. Heartbeat-Nachrichten versendet und in der Steuerungskonfiguration beim CANopen-Master die Option [Automatisch starten] aktiviert ist, wird der CANopen-Slave abhängig vom in der Node-Guarding oder Heartbeat-Nachricht empfangenen Status neu konfiguriert oder sofort wieder gestartet.

Der Master sendet Nodeguard-Nachrichten an den Slave, ...
- wenn sich der Slave im Status 4 oder höher befindet UND
- wenn Nodeguarding konfiguriert wurde.

Knoten-Status aus FB lesen:

verwendeter Funktionsbaustein	hier steht dieser Knoten-Status
CANx_MASTER_STATUS CANx_SLAVE_STATUS	Ausgang NODE_STATE_SLAVE
CANOPEN_GETSTATE	Ausgang NODESTATE

CANopen-Status des Knotens

1973

Knotenstatus nach CANopen (mit diesen Werten wird der Status auch in den entsprechenden Nachrichten vom Knoten her codiert).

Status hex	dez		CANopen-Status	Beschreibung
00	0	BOOTUP	BOOTUP-Nachricht des Knotens
04	4	STOPPED	Knoten befindet sich im Zustand STOPPED. Es findet kein Datenaustausch statt und der Knoten kann auch nicht konfiguriert werden.
05	5	OPERATIONAL	Knoten befindet sich im Zustand OPERATIONAL und nimmt am normalen Datenaustausch teil.
7F	127	PRE-OPERATIONAL	Knoten befindet sich im Zustand PRE-OPERATIONAL und kann vom Master konfiguriert werden.

Wenn Nodeguarding aktiv: das höchstwertige Status-Bit wechselt (toggelt) von Nachricht zu Nachricht.

Knoten-Status aus FB lesen:

verwendeter Funktionsbaustein	hier steht dieser Knoten-Status
CANx_MASTER_STATUS	Strukturelement LAST_STATE aus dem Array NODE_STATE_SLAVE
CANx_SLAVE_STATUS	Ausgang NODE_STATE
CANOPEN_GETSTATE	Ausgang LASTNODESTATE

8.3.4 CANopen Error-Code

Inhalt

9967

Emergency-Nachrichten

9973

Gerätefehler im Slave oder Probleme im CAN-Bus lösen Emergency-Nachrichten aus:

COB-ID	DLC	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7	Byte 8
0x080 + Node-ID	X	Error-Code		Objekt 0x1001	gerätespezifisch				

Beachten Sie die umgekehrte Byte-Reihenfolge! (⇨ Little Endian oder Intel-Format)

Übersicht CANopen-Error-Codes

8545

Error Code (hex)	Meaning / Bedeutung
00xx	Reset or no Error (Fehler rücksetzen / kein Fehler)
10xx	Generic Error (allgemeiner Fehler)
20xx	Current (Stromfehler)
21xx	Current, device input side (Stromfehler, eingangsseitig)
22xx	Current inside the device (Stromfehler im Geräteinnern)
23xx	Current, device output side (Stromfehler, ausgangsseitig)
30xx	Voltage (Spannungsfehler)
31xx	Mains Voltage
32xx	Voltage inside the device (Spannungsfehler im Geräteinnern)
33xx	Output Voltage (Spannungsfehler, ausgangsseitig)
40xx	Temperature (Temperaturfehler)
41xx	Ambient Temperature (Umgebungstemperaturfehler)
42xx	Device Temperature (Gerätetemperaturfehler)
50xx	Device Hardware (Geräte-Hardware-Fehler)
60xx	Device Software (Geräte-Software-Fehler)
61xx	Internal Software (Firmware-Fehler)
62xx	User Software (Applications-Software)
63xx	Data Set (Daten-/Parameterfehler)
70xx	Additional Modules (zusätzliche Module)
80xx	Monitoring (Überwachung)
81xx	Communication (Kommunikation)
8110	CAN Overrun-objects lost (CAN Überlauf-Datenverlust)
8120	CAN in Error Passiv Mode (CAN im Modus "fehlerpassiv")
8130	Life Guard Error or Heartbeat Error (Guarding-Fehler oder Heartbeat-Fehler)
8140	Recovered from Bus off (Bus-Off zurückgesetzt)
8150	Transmit COB-ID collision (Senden "Kollision des COB-ID")
82xx	Protocol Error (Protokollfehler)
8210	PDO not processed due to length error (PDO nicht verarbeitet, fehlerhafte Längenangabe)
8220	PDO length exceeded (PDO Längenfehler, ausgangsseitig)
90xx	External Error (Externer Fehler)
F0xx	Additional Functions (zusätzliche Funktionen)
FFxx	Device specific (gerätespezifisch)

Objekt 0x1001 (Error-Register)

8547

Dieses Objekt spiegelt den allgemeinen Fehlerzustand eines CANopen-Gerätes wider. Das Gerät ist dann als fehlerfrei anzusehen, wenn das Objekt 0x1001 keinen Fehler mehr signalisiert.

Bit	Meaning (Bedeutung)
0	Generic Error (allgemeiner Fehler)
1	Current (Stromfehler)
2	Voltage (Spannungsfehler)
3	Temperature (Temperaturfehler)
4	Communication Error (Kommunikationsfehler)
5	Device Profile specific (Geräteprofil spezifisch)
6	Reserved – always 0 (reserviert – immer 0)
7	manufacturer specific (herstellerspezifisch)

Für eine Fehlermeldung können mehrere Bits im Error-Register gleichzeitig gesetzt sein.

Beispiel: CR2033, Meldung "Leiterbruch" an Kanal 2 (→ Installationsanleitung des Geräts):

COB-ID	DLC	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7	Byte 8
0x80 + Node-ID		**00**	**FF**	**81**	**10**	**00**	00	00	00

Error-Code = 0xFF00

Error-Register = 0x81 = 0b1000 0001, besteht also aus folgenden Fehlern:
- generic error (allgemeiner Fehler)
- manufacturer specific (herstellerspezifisch)

Betroffener Kanal = 0x0010 = 0b0000 0000 0001 0000 = Kanal 2

8.4 Safety-Checklisten

Inhalt

14342

Hier finden Sie Checklisten, um mit dem SafetyController möglichst sicher zu einer sicheren Anwendung zu gelangen.

8.4.1 Checkliste: Bootprojekt erzeugen

14344

Diese Checkliste soll das Vorgehen beim Laden der für die Freigabe vorgesehenen Anwendung auf die Steuerung verständlicher machen.

Nr.	Aufgabe	prüfen	Wert
1.	Bei Zugriff über den CAN-Bus sicherstellen, dass die richtige Steuerung angesprochen wird! → Kapitel ***Regel 11 – Inbetriebnahme und Wartung der Steuerung beim Zugriff über CAN*** (→ Seite 122)	Download-ID =	
		Controller Seriennummer =	
2.	Mit dem **ifm**-Downloader (Version?) das Laufzeitsystem (Version?) auf die Steuerung laden. dazu den **ifm**-Downloader wie folgt einstellen: • [Protocol] = [CoDeSys V2.3 Tricore] • [Safety] = [with CRC] • [Options] > [File Options] > nur [Runtime System / Application Program] wählen	Downloader Version =	
		Runtime System Release =	
3.	Mit dem **ifm**-Downloader die "PLC Information" auslesen. Vergleichen: [Release] und [CRC] des vorliegenden Laufzeitsystems mit [Release] und [CRC] des Laufzeitsystems in den Freigabeunterlagen.	vorliegend: Runtime System Release =	
		vorliegend: Runtime System CRC =	
		bei Freigabe: Runtime System Release =	
		bei Freigabe: Runtime System CRC =	
		identisch mit vorliegendem Laufzeitsystem?	ja / nein
4.	Identity (→ FB ***SET_IDENTITY*** (→ Seite 379)) im Anwendungsprogramm festlegen (Name / Version) .	festgelegte ID =	
5.	Mit CODESYS die Anwendung auf die Steuerung laden: [Online] > [Bootprojekt erzeugen] Projekt in Steuerung laden	Bootprojekt erzeugt?	ja / nein
6.	Anwendung starten. Auf einwandfreien Ablauf prüfen. Wenn in Ordnung, weiter mit (7.)	Ablauf in Ordnung?	ja / nein
7.	Mit dem **ifm**-Downloader (Version?) die "PLC Information" auslesen: • Prüfen: die Identity (Name / Version?) der Anwendung • Auslesen: die [CRC] der Anwendung	Downloader Version =	
		Application Identity =	
		identisch mit festgelegter ID?	ja / nein
		Application CRC =	
8.	Archivieren: • diese Checkliste mit den notierten Werten • optional: Screenshot "PLC Information"	Screenshot existiert?	ja / nein

8.4.2 Checkliste: Anwendung mit ifm-Downloader auslesen

14345

Diese Checkliste soll das Vorgehen beim Auslesen der freigegebenen Anwendung aus der Steuerung verständlicher verständlicher machen.

Nr.	Aufgabe	prüfen	Wert
1.	Bei Zugriff über den CAN-Bus sicherstellen, dass die richtige Steuerung angesprochen wird! → Kapitel ***Regel 11 – Inbetriebnahme und Wartung der Steuerung beim Zugriff über CAN*** (→ Seite 122)	Download-ID =	
		Controller Seriennummer =	
2.	Mit dem ifm-Downloader (Version?) die Anwendung auslesen. Dazu den ifm-Downloader wie folgt einstellen: • [Protocol] = [CoDeSys V2.3 Tricore] • [Safety] = [with CRC] • [Options] > [File Options] > nur [Runtime System / Application Program] wählen Dateiname und Verzeichnis festlegen	Downloader Version = Dateiname.H86 =	
3.	Anschließend Datei mit VERIFY auf Übereinstimmung prüfen.	stimmt überein?	ja / nein
4.	Mit dem ifm-Downloader die "PLC Information" auslesen. Identity (Name / Version) und [CRC] der Anwendung mit den Angaben in der ersten Checkliste vergleichen.	vorliegend: Application Identity =	
		vorliegend: Application CRC =	
		erste Checkliste: Application Identity =	
		erste Checkliste: Application CRC =	
		identisch mit vorliegender Anwendung?	ja / nein
5.	Archivieren: • H86-Datei der Anwendung (max. 15 Zeichen) • diese Checkliste mit den notierten Werten • optional: Screenshot mit den "PLC Information"	Dateiname.H86 =	
		Screenshot existiert?	ja / nein

8.4.3 Checkliste: Anwendung mit ifm-Downloader in weitere Steuerungen laden

14346

Diese Checkliste soll das Vorgehen beim Laden der freigegebenen Anwendung auf andere Steuerungen verständlicher machen.

Nr.	Aufgabe	prüfen	Wert
1.	Bei Zugriff über den CAN-Bus sicherstellen, dass die richtige Steuerung angesprochen wird! → Kapitel ***Regel 11 – Inbetriebnahme und Wartung der Steuerung beim Zugriff über CAN*** (→ Seite 122)	Download-ID =	
		Controller Seriennummer =	
2.	Mit dem ifm-Downloader (Version?) das Laufzeitsystem (Version?) auf die Steuerung laden. dazu den ifm-Downloader wie folgt einstellen: • [Protocol] = [CoDeSys V2.3 Tricore] • [Safety] = [with CRC] • [Options] > [File Options] > nur [Runtime System / Application Program] wählen	Downloader Version =	
		Runtime System Release =	
3.	Mit dem ifm-Downloader die "PLC Information" auslesen. Vergleichen: [Release] und [CRC] des vorliegenden Laufzeitsystems mit [Release] und [CRC] des Laufzeitsystems in den Freigabeunterlagen.	vorliegend: Runtime System Release =	
		vorliegend: Runtime System CRC =	
		bei Freigabe: Runtime System Release =	
		bei Freigabe: Runtime System CRC =	
		identisch mit vorliegendem Laufzeitsystem?	ja / nein
4.	Mit dem ifm-Downloader die archivierte Anwendung (Name / Version?) auf die Steuerung laden. dazu den ifm-Downloader wie folgt einstellen: • [Protocol] = [CoDeSys V2.3 Tricore] • [Safety] = [with CRC] • [Options] > [File Options] > nur [Runtime System / Application Program] wählen	Dateiname.H86 = (max. 15 Zeichen)	
5.	Anwendung starten.	—	—
6.	Mit dem ifm-Downloader die "PLC Information" auslesen. Vergleichen: Identity (Name / Version?) und [CRC] der Anwendung mit den Angaben in der zweiten Checkliste.	vorliegend: Application Identity =	
		vorliegend: Application CRC =	
		zweite Checkliste: Application Identity =	
		zweite Checkliste: Application CRC =	
		identisch mit vorliegender Anwendung?	ja / nein
7.	Archivieren: • diese Checkliste mit den notierten Werten • optional: Screenshot "PLC Information"	Screenshot existiert?	ja / nein

9 Begriffe und Abkürzungen

A

Adresse

Das ist der „Name“ des Teilnehmers im Bus. Alle Teilnehmer benötigen eine unverwechselbare, eindeutige Adresse, damit der Austausch der Signale fehlerfrei funktioniert.

Anforderungsrate rd

Die Anforderungsrate r_d ist die Häufigkeit je Zeiteinheit von Anforderungen an eine sichere Reaktion eines →SRP/CS.

Anleitung

Übergeordnetes Wort für einen der folgenden Begriffe:
Montageanleitung, Datenblatt, Benutzerinformation, Bedienungsanleitung, Gerätehandbuch, Installationsanleitung, Onlinehilfe, Systemhandbuch, Programmierhandbuch, usw.

Anwendungsprogramm

Software, die speziell für die Anwendung vom Hersteller in die Maschine programmiert wird. Die Software enthält üblicherweise logische Sequenzen, Grenzwerte und Ausdrücke zum Steuern der entsprechenden Ein- und Ausgänge, Berechnungen und Entscheidungen.

Architektur

Spezifische Konfiguration von Hardware- und/oder Software-Elementen in einem System.

Ausfall

Ausfall ist die Beendigung der Fähigkeit einer Einheit, eine geforderte Funktion zu erfüllen.
Nach einem Ausfall hat die Einheit einen →Fehler. Der Ausfall ist ein Ereignis, der Fehler ein Zustand.
Der so definierte Begriff kann nicht auf Einheiten angewendet werden, die nur aus Software bestehen.

Ausfall, gefahrbringend

Ein gefahrbringender Ausfall hat das Potential, das →SRP/CS in einen gefährlichen Zustand oder eine Fehlfunktion zu bringen. Ob dieses Potential bemerkt werden kann oder nicht, hängt von der Architektur des Systems ab. In einem redundanten System wird ein gefährlicher Hardware-Ausfall weniger wahrscheinlich zu einem gefährlichen Ausfall des Gesamtsystems führen.

Ausfall, systematischer

Ein systematischer Ausfall ist ein Ausfall mit deterministischem (= nicht zufälligem) Bezug zu einer bestimmten Ursache. Der systematische Ausfall kann nur beseitigt werden durch Änderung des Entwurfs oder des Herstellprozesses, Betriebsverfahren, der Dokumentation oder zugehörigen Faktoren.

Eine Instandsetzung ohne Änderung des Systems wird den Grund des systematischen Ausfalls in der Regel nicht beseitigen.

B

Baud

Baud, Abk.: Bd = Maßeinheit für die Geschwindigkeit bei der Datenübertragung. Baud ist nicht zu verwechseln mit "bits per second" (bps, Bit/s). Baud gibt zwar die Anzahl von Zustandsänderungen (Schritte, Takte) pro Sekunde auf einer Übertragungsstrecke an. Aber es ist nicht festgelegt, wie viele Bits pro Schritt übertragen werden. Der Name Baud geht auf den französischen Erfinder J. M. Baudot zurück, dessen Code für Telexgeräte verwendet wurde.
1 MBd = 1024 x 1024 Bd = 1 048 576 Bd

Bestimmungsgemäße Verwendung

Das ist die Verwendung eines Produkts in Übereinstimmung mit den in der Anleitung bereitgestellten Informationen.

Betriebsdauer, mittlere

Mean **t**ime **b**etween **f**ailures (MTBF) = mittlere Betriebsdauer zwischen Ausfällen.
Ist der Erwartungswert der Betriebsdauer zwischen zwei aufeinanderfolgenden Ausfällen von Einheiten, die instand gesetzt werden.
(!) Für Einheiten, die NICHT instandgesetzt werden, ist der Erwartungswert (Mittelwert) der Verteilung von Lebensdauern die mittlere Lebensdauer →MTTF.

Bootloader

Im Auslieferungszustand enthalten ecomat*mobile*-Controller nur den Bootloader.
Der Bootloader ist ein Startprogramm, mit dem das Laufzeitsystem und das Anwendungsprogramm auf dem Gerät nachgeladen werden können.
Der Bootloader enthält Grundroutinen...

- zur Kommunikation der Hardware-Module untereinander,
- zum Nachladen des Laufzeitsystems.

Der Bootloader ist das erste Software-Modul, das im Gerät gespeichert sein muss.

Bus

Serielle Datenübertragung mehrerer Teilnehmer an derselben Leitung.

C

CAN

CAN = **C**ontroller **A**rea **N**etwork
CAN gilt als Feldbussystem für größere Datenmengen, das prioritätengesteuert arbeitet. Es gibt mehrere höhere Protokolle, die auf CAN aufsetzen, z. B. 'CANopen' oder 'J1939'.

CAN-Stack

CAN-Stack = Software-Komponente, die sich um die Verarbeitung von CAN-Telegramme kümmert.

CCF

Common **c**ause **f**ailure = Ausfall in Folge von gemeinsamer Ursache
Ausfälle verschiedener Einheiten aufgrund eines gemeinsamen Ereignisses, wobei diese Ausfälle nicht auf gegenseitige Ursachen beruhen.

CiA

CiA = CAN in Automation e.V.
Anwender- und Herstellerorganisation in Erlangen, Deutschland. Definitions- und Kontrollorgan für das CANopen-Protokoll.
Homepage → www.can-cia.org

CiA DS 304

DS = **D**raft **S**tandard
CANopen-Geräteprofil für sichere Kommunikation

CiA DS 401

DS = **D**raft **S**tandard
CANopen-Geräteprofil für digitale und analoge E/A-Baugruppen

CiA DS 402

DS = **D**raft **S**tandard
CANopen-Geräteprofil für Antriebe

CiA DS 403

DS = **D**raft **S**tandard
CANopen-Geräteprofil für Bediengeräte

CiA DS 404

DS = **D**raft **S**tandard
CANopen-Geräteprofil für Messtechnik und Regler

CiA DS 405

DS = **D**raft **S**tandard
CANopen-Spezifikation der Schnittstelle zu programmierbaren Steuerungen (IEC 61131-3)

CiA DS 406

DS = **D**raft **S**tandard
CANopen-Geräteprofil für Drehgeber / Encoder

CiA DS 407

DS = **D**raft **S**tandard
CANopen-Anwendungsprofil für den öffentlichen Nahverkehr

COB-ID

COB = **C**ommunication **Ob**ject = Kommunikationsobjekt
ID = **Id**entifier = Kennung
ID eines CANopen-Kommunikationsobjekts
Entspricht dem Identifier der CAN-Nachricht, mit der das Kommunikationsobjekt über den CAN-Bus gesendet wird.

CODESYS

CODESYS® ist eingetragene Marke der 3S – Smart Software Solutions GmbH, Deutschland.
'CODESYS for Automation Alliancetm' vereinigt Firmen der Automatisierungsindustrie, deren Hardware-Geräte alle mit dem weit verbreiteten IEC 61131-3 Entwicklungswerkzeug CODESYS® programmiert werden. Homepage → www.codesys.com

CRC

CRC = **C**yclic **R**edundancy **C**heck = zyklische Redundanzprüfung.
CRC ist ein Verfahren aus der Informationstechnik zur Bestimmung eines Prüfwerts für Daten, um Fehler bei der Übertragung oder Duplizierung von Daten erkennen zu können.
Vor Beginn der Übertragung eines Blocks der Daten wird ein CRC-Wert berechnet. Nach Abschluss der Transaktion wird am Zielort der CRC-Wert erneut berechnet. Anschließend werden diese beiden Prüfwerte verglichen.

CSV-Datei

CSV = **C**omma **S**eparated **V**alues (auch: **C**haracter **S**eparated **V**alues)
Eine CSV-Datei ist eine Textdatei zur Speicherung oder zum Austausch einfach strukturierter Daten.
Die Dateinamen-Erweiterung lautet `.csv`.

Beispiel: Quell-Tabelle mit Zahlenwerten:

Wert 1.0	Wert 1.1	Wert 1.2	Wert 1.3
Wert 2.0	Wert 2.1	Wert 2.2	Wert 2.3
Wert 3.0	Wert 3.1	Wert 3.2	Wert 3.3

Daraus entsteht folgende CSV-Datei:

```
Wert 1.0;Wert 1.1;Wert 1.2;Wert 1.3
Wert 2.0;Wert 2.1;Wert 2.2;Wert 2.3
Wert 3.0;Wert 3.1;Wert 3.2;Wert 3.3
```

D

Datentyp

Abhängig vom Datentyp können unterschiedlich große Werte gespeichert werden.

Datentyp	min. Wert	max. Wert	Größe im Speicher
BOOL	FALSE	TRUE	8 Bit = 1 Byte
BYTE	0	255	8 Bit = 1 Byte
WORD	0	65 535	16 Bit = 2 Bytes
DWORD	0	4 294 967 295	32 Bit = 4 Bytes
SINT	-128	127	8 Bit = 1 Byte
USINT	0	255	8 Bit = 1 Byte
INT	-32 768	32 767	16 Bit = 2 Bytes
UINT	0	65 535	16 Bit = 2 Bytes
DINT	-2 147 483 648	2 147 483 647	32 Bit = 4 Bytes
UDINT	0	4 294 967 295	32 Bit = 4 Bytes
REAL	$-3{,}402823466 \cdot 10^{38}$	$3{,}402823466 \cdot 10^{38}$	32 Bit = 4 Bytes
ULINT	0	18 446 744 073 709 551 615	64 Bit = 8 Bytes
STRING			number of char. + 1

DC

Diagnostic **C**overage = Diagnose-Deckungsgrad
Der Diagnose-Deckungsgrad ist das Maß für die Wirksamkeit der →Diagnose als Verhältnis der Ausfallrate der bemerkten gefahrbringenden Ausfälle und der Ausfallrate der gesamten gefahrbringenden Ausfälle:

Formel: DC = Ausfallrate bemerkte gefahrbringende Ausfälle / Ausfallrate gesamte gefahrbringende Ausfälle

Bezeichnung	Bereich
kein	DC < 60 %
niedrig	60 % ≤ DC < 90 %
mittel	90 % ≤ DC < 99 %
hoch	99 % ≤ DC

Tabelle: Diagnose-Deckungsgrad DC

Für die in der Tabelle gezeigten Grenzwerte wird eine Genauigkeit von 5 % angenommen.
Der Diagnose-Deckungsgrad kann für das gesamte sicherheitsgerichtete System ermittelt werden oder nur für Teile des sicherheitsgerichteten Systems.

DC

Direct **C**urrent = Gleichstrom

Diagnose

Bei der Diagnose wird der "Gesundheitszustand" des Gerätes geprüft. Es soll festgestellt werden, ob und gegebenenfalls welche →Fehler im Gerät vorhanden sind.

Je nach Gerät können auch die Ein- und Ausgänge auf einwandfreie Funktion überwacht werden:
- Drahtbruch,
- Kurzschluss,
- Wert außerhalb des Sollbereichs.

Zur Diagnose können Konfigurations-Dateien herangezogen werden, die während des "normalen" Betriebs des Gerätes erzeugt wurden.
Der korrekte Start der Systemkomponenten wird während der Initialisierungs- und Startphase überwacht.
Zur weiteren Diagnose können auch Selbsttests durchgeführt werden.

Diagnose-Deckungsgrad

→ ***DC*** (→ Seite 468)

Dither

to dither (engl.) = schwanken / zittern.
Dither ist ein Bestandteil der →PWM-Signale zum Ansteuern von Hydraulik-Ventilen. Für die elektromagnetischen Antriebe von Hydraulik-Ventilen hat sich herausgestellt, dass sich die Ventile viel besser regeln lassen, wenn das Steuersignal (PWM-Impulse) mit einer bestimmten Frequenz der PWM-Frequenz überlagert wird. Diese Dither-Frequenz muss ein ganzzahliger Teil der PWM-Frequenz sein.

diversitär

Unter Diversität (Vielfalt) versteht man in der Technik eine Strategie zur Erhöhung der Ausfallsicherheit.
Dabei werden Systeme →redundant ausgelegt, allerdings werden bewusst verschiedene Realisierungen und keine baugleichen Einzelsysteme verwendet. Man geht davon aus, dass Systeme, die das Gleiche leisten, aber unterschiedlich realisiert sind, auch gegen unterschiedliche Störungen empfindlich oder unempfindlich sind und daher möglichst nicht alle gleichzeitig ausfallen.
Die konkrete Realisierung kann je nach Einsatzgebiet und geforderter Sicherheit unterschiedlich aussehen:

- Verwendung von Bauteilen verschiedener Hersteller,
- Nutzung unterschiedlicher Protokolle zur Steuerung von Geräten,
- Verwendung komplett unterschiedlicher Technologien, beispielsweise einer elektrischen und einer pneumatischen Steuerung,
- Verwendung unterschiedlicher Messmethoden (Strom, Spannung),
- zwei Kanäle mit gegenläufigen Werteverläufen:
  Kanal A: 0...100 %
  Kanal B: 100...0 %

DLC

Data **L**ength **C**ode = bei CANopen die Anzahl der Daten-Bytes in einer Nachricht.
Für →SDO: DLC = 8

DRAM

DRAM = **D**ynamic **R**andom **A**ccess **M**emory.
Technologie für einen elektronischen Speicherbaustein mit wahlfreiem Zugriff (Random Access Memory, RAM). Das speichernde Element ist dabei ein Kondensator, der entweder geladen oder entladen ist. Über einen Schalttransistor wird er zugänglich und entweder ausgelesen oder mit neuem Inhalt beschrieben. Der Speicherinhalt ist flüchtig: die gespeicherte Information geht bei fehlender Betriebsspannung oder zu später Wiederauffrischung verloren.

DTC

DTC = **D**iagnostic **T**rouble **C**ode = Fehler-Code
Beim Protokoll J1939 werden Störungen und Fehler über zugeordnete Nummern – den DTCs – verwaltet und gemeldet.

E

ECU

(1) **E**lectronic **C**ontrol **U**nit = Steuergerät oder Mikrocontroller
(2) **E**ngine **C**ontrol **U**nit = Steuergerät eines Motors

EDS-Datei

EDS = **E**lectronic **D**ata **S**heet = elektronisch hinterlegtes Datenblatt, z.B. für:
• Datei für das Objektverzeichnis im CANopen-Master,
• CANopen-Gerätebeschreibungen.
Via EDS können vereinfacht Geräte und Programme ihre Spezifikationen austauschen und gegenseitig berücksichtigen.

Embedded Software

System-Software, Grundprogramm im Gerät, praktisch das →Laufzeitsystem.
Die Firmware stellt die Verbindung her zwischen der Hardware des Gerätes und dem Anwendungsprogramm. Die Firmware wird vom Hersteller der Steuerung als Teil des Systems geliefert und kann vom Anwender nicht verändert werden.

EMCY

Abkürzung für Emergency (engl.) = Notfall
Nachricht im CANopen-Protokoll, mit der Fehler gemeldet werden.

EMV

EMV = **E**lektro-**M**agnetische **V**erträglichkeit.
Gemäß der EG-Richtlinie (2004/108/EG) zur elektromagnetischen Verträglichkeit (kurz EMV-Richtlinie) werden Anforderungen an die Fähigkeit von elektrischen und elektronischen Apparaten, Anlagen, Systemen oder Bauteilen gestellt, in der vorhandenen elektromagnetischen Umwelt zufriedenstellend zu arbeiten. Die Geräte dürfen ihre Umgebung nicht stören und dürfen sich von äußerlichen elektromagnetischen Störungen nicht ungünstig beeinflussen lassen.

Erstfehler-Eintrittszeit

Das ist die Zeit bis zum ersten Versagen eines Sicherheitselements.
Innerhalb eines spezifizierten Zeitraums überprüft das →Laufzeitsystem mittels interner Überwachungs- und Testroutinen die Steuerung.
Diese "Testzykluszeit" muss kleiner sein als die statistische Erstfehler-Eintrittszeit für die Anwendung.

Ethernet

Ethernet ist eine weit verbreitete, herstellerneutrale Netzwerktechnologie, mit der Daten mit einer Geschwindigkeit von 10 bis 10 000 Millionen Bit pro Sekunde (Mbps) übertragen werden können. Ethernet gehört zu der Familie der sogenannten „bestmöglichen Datenübermittlung“ auf einem nicht exklusiven Übertragungsmedium. 1972 entwickelt, wurde das Konzept 1985 als IEEE 802.3 spezifiziert.

EUC

EUC = **E**quipment **U**nder **C**ontrol (kontrollierte Einrichtung).
EUC ist eine Einrichtung, Maschine, Gerät oder Anlage, verwendet zur Fertigung, Stoffumformung, zum Transport, zu medizinischen oder anderen Tätigkeiten (→ IEC 61508-4, Abschnitt 3.2.3). Das EUC umfasst also alle Einrichtungen, Maschinen, Geräte oder Anlagen, die →Gefährdungen verursachen können und für die sicherheitsgerichtete Systeme erforderlich sind.
Falls eine vernünftigerweise vorhersehbare Aktivität oder Inaktivität zu durch das EUC verursachten Gefährdungen mit unvertretbarem Risiko führt, sind Sicherheitsfunktionen erforderlich, um einen sicheren Zustand für das EUC zu erreichen oder aufrecht zu erhalten. Diese Sicherheitsfunktionen werden durch ein oder mehrere sicherheitsgerichtete Systeme ausgeführt.

F

Fehlanwendung

Das ist die Verwendung eines Produkts in einer Weise, die vom Konstrukteur nicht vorgesehen ist.
Eine Fehlanwendung führt meist zu einer →Gefährdung von Personen oder Sachen.
Vor vernünftigerweise, vorhersehbaren Fehlanwendungen muss der Hersteller des Produkts in seinen Benutzerinformationen warnen.

Fehler

Ein Fehler ist die Unfähigkeit einer Einheit, eine geforderte Funktion auszuführen.
Kein Fehler ist diese Unfähigkeit während vorbeugender Wartung oder anderer geplanter Handlungen oder aufgrund des Fehlers externer Mittel.
Ein Fehler ist oft das Resultat eines Ausfalls der Einheit selbst, kann aber ohne vorherigen Ausfall bestehen.
In der ISO 13849-1 ist mit "Fehler" der "zufällige Fehler" gemeint.

FiFo

FIFO (**F**irst **I**n, **F**irst **O**ut) = Arbeitsweise des Stapelspeichers: Das Datenpaket, das zuerst in den Stapelspeicher geschrieben wurde, wird auch als erstes gelesen. Pro Identifier steht ein solcher Zwischenspeicher (als Warteschlange) zur Verfügung.

Flash-Speicher

Flash-ROM (oder Flash-EPROM oder Flash-Memory) kombiniert die Vorteile von Halbleiterspeicher und Festplatten. Die Daten werden allerdings wie bei einer Festplatte blockweise in Datenblöcken zu 64, 128, 256, 1024, ... Byte zugleich geschrieben und gelöscht.

Vorteile von Flash-Speicher

- Die gespeicherten Daten bleiben auch bei fehlender Versorgungsspannung erhalten.
- Wegen fehlender beweglicher Teile ist Flash geräuschlos, unempfindlich gegen Erschütterungen und magnetische Felder.

Nachteile von Flash-Speicher

- Begrenzte Zahl von Schreib- bzw. Löschvorgängen, die eine Speicherzelle vertragen kann:
  • Multi-Level-Cells: typ. 10 000 Zyklen
  • Single-Level-Cells: typ. 100 000 Zyklen
- Da ein Schreibvorgang Speicherblöcke zwischen 16 und 128 kByte gleichzeitig beschreibt, werden auch Speicherzellen beansprucht, die gar keiner Veränderung bedürfen.

FMEA

FMEA = **F**ailure **M**ode and **E**ffects **A**nalysis = **F**ehler-**M**öglichkeits- und **E**influss-**A**nalyse.
Methode der Zuverlässigkeitstechnik, um potenzielle Schwachstellen zu finden. Im Rahmen des Qualitäts- oder Sicherheitsmanagements wird die FMEA zur Fehlervermeidung und Erhöhung der technischen Zuverlässigkeit vorbeugend eingesetzt.

FRAM

FRAM, oder auch FeRAM, bedeutet **Fe**rroelectric **R**andom **A**ccess **M**emory. Der Speicher- und Löschvorgang erfolgt durch eine Polarisationsänderung in einer ferroelektrischen Schicht.
Vorteile von FRAM gegenüber herkömmlichen Festwertspeichern:
• nicht flüchtig,
• kompatibel zu gängigen EEPROMs, jedoch:
• Zugriffszeit ca. 100 ns,
• fast unbegrenzt viele Zugriffszyklen möglich.

Funktionale Sicherheit

Teil der Gesamtsicherheit, bezogen auf das →EUC und das EUC-Leit- oder Steuerungssystem, die von der korrekten Funktion des elektrischen oder elektronischen sicherheitsgerichteten Systems, sicherheitsgerichteten Systemen anderer Technologien und externer Einrichtungen zur Risikominderung abhängt.

G

Gebrauchsdauer Tm

Die Gebrauchsdauer T_M ist der Zeitraum, der die vorgegebene Verwendung der SRP/CS abdeckt.

Gefährdung

Mit Gefährdung bezeichnet man eine potentielle Schadensquelle.
Man unterscheidet den Ursprung der Gefährdung, z.B.:
- mechanische Gefährdung,
- elektrische Gefährdung,

oder die Art des zu erwartenden Schadens, z.B.:
- Gefährdung durch elektrischen Schlag,
- Gefährdung durch Schneiden,
- Gefährdung durch Vergiftung.

Die Gefährdung im Sinne dieser Definition ist bei der bestimmungsgemäßen Verwendung der Maschine entweder dauerhaft vorhanden, z.B.:
- Bewegung von gefährdenden beweglichen Teilen,
- Lichtbogen beim Schweißen,
- ungesunde Körperhaltung,
- Geräusch-Emission,
- hohe Temperatur,

oder die Gefährdung kann unerwartet auftreten, z.B.:
- Explosion,
- Gefährdung durch Quetschen als Folge eines unbeabsichtigten / unerwarteten Anlaufs,
- Herausschleudern als Folge eines Bruchs,
- Stürzen als Folge von Geschwindigkeitsänderung.

H

Heartbeat

Heartbeat (engl.) = Herzschlag.
Die Teilnehmer senden regelmäßig kurze Signale. So können die anderen Teilnehmer prüfen, ob ein Teilnehmer ausgefallen ist.

HFT

HFT = Hardware-Fehler-Toleranz (engl. Hardware Fault Tolerance) ist in der Norm IEC 61508 eine Kennzahl zur Beschreibung von Systemen mit sicherheitsgerichteter Funktion.

- Eine HFT = N gibt an, dass N + 1 Hardware-Fehler, ungünstig verteilt, zum Verlust der Sicherheitsfunktion führen können.
- Je größer der Wert für HFT ist, desto besser ist das Gerät für hohe Sicherheitsanforderungen geeignet.

HMI

HMI = **H**uman **M**achine **I**nterface = Mensch-Maschine-Schnittstelle

I

ID – Identifier

ID = **Id**entifier = Kennung

Name zur Unterscheidung der an einem System angeschlossenen Geräte / Teilnehmer oder der zwischen den Teilnehmern ausgetauschten Nachrichtenpakete.

IEC 61131

Norm: Grundlagen Speicherprogrammierbarer Steuerungen
- Teil 1: Allgemeine Informationen
- Teil 2: Betriebsmittelanforderungen und Prüfungen
- Teil 3: Programmiersprachen
- Teil 5: Kommunikation
- Teil 7: Fuzzy-Control-Programmierung

IEC 61508

Norm: Funktionale Sicherheit sicherheitsbezogener elektrischer / elektronischer / programmierbarer elektronischer Systeme

IEC-User-Zyklus

IEC-User-Zyklus = SPS-Zyklus im CODESYS-Anwendungsprogramm.

IP-Adresse

IP = **I**nternet **P**rotocol = Internet-Protokoll.
Die IP-Adresse ist eine Nummer, die zur eindeutigen Identifizierung eines Internet-Teilnehmers notwendig ist. Zur besseren Übersicht wird die Nummer in 4 dezimalen Werten geschrieben, z. B. 127.215.205.156.

ISO 11898

Norm: Straßenfahrzeuge – CAN-Protokoll
- Teil 1: Bit-Übertragungsschicht und physikalische Zeichenabgabe
- Teil 2: High-speed medium access unit
- Teil 3: Fehlertolerante Schnittstelle für niedrige Geschwindigkeiten
- Teil 4: Zeitgesteuerte Kommunikation
- Teil 5: High-speed medium access unit with low-power mode

ISO 11992

Norm: Straßenfahrzeuge – Austausch von digitalen Informationen über elektrische Verbindungen zwischen Zugfahrzeugen und Anhängefahrzeugen
- Teil 1: Bit-Übertragungsschicht und Sicherungsschicht
- Teil 2: Anwendungsschicht für die Bremsausrüstung
- Teil 3: Anwendungsschicht für andere als die Bremsausrüstung
- Teil 4: Diagnose

ISO 13849

Norm: Sicherheit von Maschinen – Sicherheitsbezogene Teile von Steuerungen
- Teil 1: Allgemeine Gestaltungsleitsätze
- Teil 2: Validierung

ISO 16845

Norm: Straßenfahrzeuge – Steuergerätenetz (CAN) – Prüfplan zu Konformität

J

J1939

→ SAE J1939

K

Kategorie (Cat.)

Einstufung der sicherheitsrelevanten Teile einer Steuerung bezüglich ihres Widerstandes gegen →Fehler und ihres nachfolgenden Verhaltens bei einem Fehler. Diese Sicherheit wird erreicht durch die Struktur der Anordnung der Teile, die Fehlererkennung und/oder ihre Zuverlässigkeit (→ ISO 13849).

Klemme 15

Klemme 15 ist in Fahrzeugen die vom Zündschloss geschaltete Plusleitung.

L

Laufzeitsystem

Grundprogramm im Gerät, stellt die Verbindung her zwischen der Hardware des Gerätes und dem Anwendungsprogramm.
→ Kapitel ***Software-Module für das Gerät*** (→ Seite 159)

Lebensdauer, mittlere

Mean **t**ime **t**o **d**angerous **f**ailure = erwartete mittlere Dauer bis zum gefahrbringenden Ausfall.

Bezeichnung	Bereich
niedrig	3 Jahre $\leq$ $MTTF_d$ < 10 Jahre
mittel	10 Jahre $\leq$ $MTTF_d$ < 30 Jahre
hoch	30 Jahre $\leq$ $MTTF_d$ $\leq$ 100 Jahre

Tabelle: Mittlere Zeit jedes Kanals bis zum gefahrbringenden Ausfall $MTTF_d$

LED

LED = **L**ight **E**mitting **D**iode = Licht aussendende Diode.
Leuchtdiode, auch Luminiszenzdiode, ein elektronisches Element mit hoher, farbiger Leuchtkraft auf kleinem Volumen bei vernachlässigbarer Verlustleistung.

Link

Ein Link ist ein Querverweis zu einer anderen Stelle im Dokument oder auf ein externes Dokument.

LSB

Least **S**ignificant **B**it/Byte = Niederwertigstes Bit/Byte in einer Reihe von Bit/Bytes.

M

MAC-ID

MAC = **M**anufacturer‘s **A**ddress **C**ode
= Hersteller-Seriennummer.
→ID = **Id**entifier = Kennung
Jede Netzwerkkarte verfügt über eine so genannte MAC-Adresse, ein unverwechselbarer, auf der ganzen Welt einzigartiger Zahlencode – quasi eine Art Seriennummer. So eine MAC-Adresse ist eine Aneinanderreihung von 6 Hexadezimalzahlen, etwa "00-0C-6E-D0-02-3F".

Master

Wickelt die komplette Organisation auf dem →Bus ab. Der Master entscheidet über den zeitlichen Buszugriff und fragt die →Slaves zyklisch ab.

MMI

MMI = **M**ensch-**M**aschine-**I**nterface
→ ***HMI*** (→ Seite 473)

MRAM

MRAM = **M**agnetoresistive **R**andom **A**ccess **M**emory
Die Informationen werden mit magnetischen Ladungselementen gespeichert. Dabei wird die Eigenschaft bestimmter Materialien ausgenutzt, die ihren elektrischen Widerstand unter dem Einfluss magnetischer Felder ändern.
Vorteile von MRAM gegenüber herkömmlichen Festwertspeichern:
- nicht flüchtig (wie FRAM), jedoch:
- Zugriffszeit nur ca. 35 ns,
- unbegrenzt viele Zugriffszyklen möglich.

MSB

Most **S**ignificant **B**it/Byte = Höchstwertiges Bit/Byte einer Reihe von Bits/Bytes.

MTBF

Mean **t**ime **b**etween **f**ailures (MTBF) = mittlere Betriebsdauer zwischen Ausfällen.
Ist der Erwartungswert der Betriebsdauer zwischen zwei aufeinanderfolgenden Ausfällen von Einheiten, die instand gesetzt werden.
(!) Für Einheiten, die NICHT instandgesetzt werden, ist der Erwartungswert (Mittelwert) der Verteilung von Lebensdauern die mittlere Lebensdauer →MTTF.

MTTF

Mean **t**ime **t**o **f**ailure (MTTF) = mittlere Dauer bis zum Ausfall oder: mittlere Lebensdauer.

MTTFd

Mean **t**ime **t**o **d**angerous **f**ailure = erwartete mittlere Dauer bis zum gefahrbringenden Ausfall.

Bezeichnung	Bereich
niedrig	3 Jahre $\leq$ $MTTF_d$ < 10 Jahre
mittel	10 Jahre $\leq$ $MTTF_d$ < 30 Jahre
hoch	30 Jahre $\leq$ $MTTF_d$ $\leq$ 100 Jahre

Tabelle: Mittlere Zeit jedes Kanals bis zum gefahrbringenden Ausfall $MTTF_d$

Muting

Mit Muting bezeichnet man die vorübergehende und automatische Unterdrückung einer →Sicherheitsfunktion durch das →SRP/CS.
Beispiel: Der Sicherheits-Lichtvorhang ist überbrückt, wenn die schließenden Werkzeuge unter einen fingersicheren Abstand zueinander gelangt sind. Die bedienende Person kann nun gefahrlos an die Maschine herantreten und das Werkstück führen.

N

NMT

NMT = **N**etwork **M**anagemen**t** = Netzwerk-Verwaltung (hier: im CANopen-Protokoll).
Der NMT-Master steuert die Betriebszustände der NMT-Slaves.

Node

Node (engl.) = Knoten. Damit ist ein Teilnehmer im Netzwerk gemeint.

Node Guarding

Node (engl.) = Knoten, hier: Netzwerkteilnehmer
Guarding (engl.) = Schutz
Parametrierbare, zyklische Überwachung von jedem entsprechend konfigurierten →Slave. Der →Master prüft, ob die Slaves rechtzeitig antworten. Die Slaves prüfen, ob der Master regelmäßig anfragt. Somit können ausgefallene Netzwerkteilnehmer schnell erkannt und gemeldet werden.

O

Obj / Objekt

Oberbegriff für austauschbare Daten / Botschaften innerhalb des CANopen-Netzwerks.

Objektverzeichnis

Das **Obj**ekt**v**erzeichnis OBV enthält alle CANopen-Kommunikationsparameter eines Gerätes, sowie gerätespezifische Parameter und Daten.

OBV

Das **Obj**ekt**v**erzeichnis OBV enthält alle CANopen-Kommunikationsparameter eines Gerätes, sowie gerätespezifische Parameter und Daten.

OPC

OPC = **O**LE for **P**rocess **C**ontrol = Objektverknüpfung und -einbettung für Prozesssteuerung
Standardisierte Software-Schnittstelle zur herstellerunabhängigen Kommunikation in der Automatisierungstechnik
OPC-Client (z.B. Gerät zum Parametrieren oder Programmieren) meldet sich nach dem Anschließen am OPC-Server (z.B. Automatisierungsgerät) automatisch bei diesem an und kommuniziert mit ihm.

operational

Operational (engl.) = betriebsbereit
Betriebszustand eines CANopen-Teilnehmers. In diesem Modus können →SDOs, →NMT-Kommandos und →PDOs übertragen werden.

OSSD

OSSD = **O**utput **S**ignal **S**witching **D**evice
= Ausgangssignal eines Sicherheitsschaltgerätes, z.B.: SafetySwitch, AS-i Sicherheitsmonitor.

P

PC-Karte

→ PCMCIA-Karte

PCMCIA-Karte

PCMCIA = Personal Computer Memory Card International Association, ein Standard für Erweiterungskarten mobiler Computer.
Seit der Einführung des Cardbus-Standards 1995 werden PCMCIA-Karten auch als PC-Karte (engl.: PC Card) bezeichnet.

PDM

PDM = **P**rocess and **D**ialog **M**odule = **P**rozess- und **D**ialog-**M**onitor.
Gerät zur Kommunikation des Bedieners mit der Maschine / Anlage.

PDO

PDO = **P**rocess **D**ata **O**bject = Nachrichten-Objekt mit Prozessdaten.
Die zeitkritischen Prozessdaten werden mit Hilfe der "Process Data Objects" (PDOs) übertragen. Die PDOs können beliebig zwischen den einzelnen Knoten ausgetauscht werden (PDO-Linking).
Zusätzlich wird festgelegt, ob der Datenaustausch ereignisgesteuert (asynchron) oder synchronisiert erfolgen soll. Je nach der Art der zu übertragenden Daten kann die richtige Wahl der Übertragungsart zu einer erheblichen Entlastung des →CAN-Bus führen.
Dem Protokoll entsprechend, sind diese Dienste nicht bestätigte Dienste: es gibt keine Kontrolle, ob die Nachricht auch beim Empfänger ankommt. Netzwerkvariablen-Austausch entspricht einer "1-zu-n-Verbindung" (1 Sender zu n Empfängern).

PDU

PDU = **Pro**tocol **D**ata **U**nit = Protokoll-Daten-Einheit.
Die PDU ist ein Begriff aus dem →CAN-Protokoll →SAE J1939. Sie bezeichnet einen Bestandteil der Ziel- oder Quelladresse.

Performance-Level

Performance-**L**evel
Ist nach →ISO 13849-1 eine Einstufung (PL a...e) der Fähigkeit von sicherheitsrelevanten Teilen einer Steuerung, eine →Sicherheitsfunktion unter vorhersehbaren Bedingungen auszuführen.
→ Kapitel ***Performance-Level PL***
→ Kapitel ***Erforderlichen PL (=PLr) mittels Risikograf herleiten*** (→ Seite 25)

PES

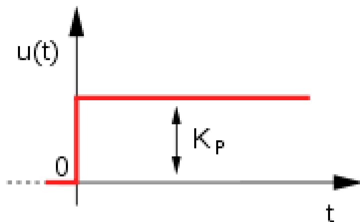
Programable **e**lectronic **s**ystem = Programmierbares elektronisches System ...
• zur Steuerung, zum Schutz oder zur Überwachung,
• auf der Basis einer oder mehrerer programmierbarer Geräte,
• einschließlich aller Elemente dieses Systems, wie Ein- und Ausgabegeräte.

PGN

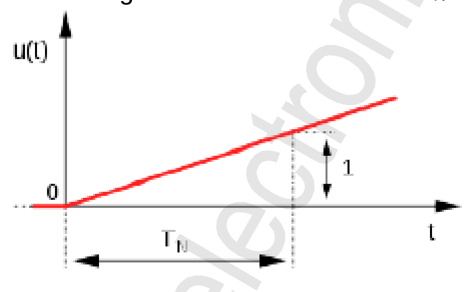
PGN = **P**arameter **G**roup **N**umber = Parameter-Gruppennummer
PGN = PDU Format (PF) + PDU Source (PS)
Die Parameter-Gruppennummer ist ein Begriff aus dem →CAN-Protokoll →SAE J1939. Sie fasst die Teiladressen PF und PS zusammen.

PID-Regler

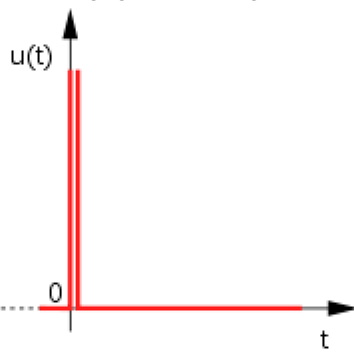
- **P = Proportional-Anteil**
  Der P-Regler besteht ausschließlich aus einem proportionalen Anteil der Verstärkung K_p. Mit seinem Ausgangssignal ist er proportional dem Eingangssignal.



- **I = Integral-Anteil**
  Ein I-Regler wirkt durch zeitliche Integration der Regelabweichung auf die Stellgröße mit der Gewichtung durch die Nachstellzeit T_N



- **D = Differential-Anteil**
  Der D-Regler reagiert nicht auf die Regelabweichung, sondern nur auf deren Änderungsgeschwindigkeit.



Piktogramm

Piktogramme sind bildhafte Symbole, die eine Information durch vereinfachte grafische Darstellung vermitteln (→ Kapitel ***Was bedeuten die Symbole und Formatierungen?*** (→ Seite 9)).

PL

Performance-**L**evel
Ist nach →ISO 13849-1 eine Einstufung (PL a...e) der Fähigkeit von sicherheitsrelevanten Teilen einer Steuerung, eine →Sicherheitsfunktion unter vorhersehbaren Bedingungen auszuführen.
→ Kapitel ***Performance-Level PL***
→ Kapitel ***Erforderlichen PL (=PLr) mittels Risikograf herleiten*** (→ Seite 25)

PLr

Mit dem "erforderlichen Performance-Level" PL_r wird nach →ISO 13849 die erforderliche Risikominderung für jede →Sicherheitsfunktion erreicht.
Für jede gewählte Sicherheitsfunktion, die durch ein →SRP/CS ausgeführt wird, muss ein PL_r festgelegt und dokumentiert werden. Die Bestimmung des PL_r ist das Ergebnis der →Risikobeurteilung, bezogen auf den Anteil der Risikominderung durch die sicherheitsrelevanten Teile der Steuerung.

Pre-Op

Pre-Op = PRE-OPERATIONAL mode (engl.) = Zustand vor 'betriebsbereit'.
Betriebszustand eines CANopen-Teilnehmers. Nach dem Einschalten der Versorgungsspannung geht jeder Teilnehmer automatisch in diesem Zustand. Im CANopen-Netz können in diesem Modus nur →SDOs und →NMT-Kommandos übertragen werden, jedoch keine Prozessdaten.

Programmiersprache, sicherheitsrelevant

4053

Für sicherheitsrelevante Anwendungen sollten nur folgende Programmiersprachen verwendet werden:

- Programmiersprache mit eingeschränktem Sprachumfang (LVL = **l**imited **v**ariability **l**anguage).
  In →CODESYS sind das Kontaktplan KOP (Ladder Diagram LD) und Funktionsplan FUP (Function block diagram FBD).
- Programmiersprache mit nicht eingeschränktem Sprachumfang (FVL = **f**ull **v**ariability **l**anguage), bei Anwendung von Programmierrichtlinien (Coding Rules).
  Dazu gehören z.B. C, C++, Assembler. In CODESYS ist das Strukturierter Text (ST).

Prozessabbild

Mit Prozessabbild bezeichnet man den Zustand der Ein- und Ausgänge, mit denen die SPS innerhalb eines →Zyklusses arbeitet.

- Am Zyklus-Beginn liest die SPS die Zustände aller Eingänge in das Prozessabbild ein. Während des Zyklusses kann die SPS Änderungen an den Eingängen nicht erkennen.
- Im Laufe des Zyklusses werden die Ausgänge nur virtuell (im Prozessabbild) geändert.
- Am Zyklus-Ende schreibt die SPS die virtuellen Ausgangszustände auf die realen Ausgänge.

Prozesssicherheitszeit

Das ist die maximale Zeit, die zwischen dem Entstehen eines Fehlers und der Einnahme des sicheren Zustandes in der Anwendung vergehen darf, ohne dass eine Gefahr für Personen zu befürchten ist.
Dabei sind die Sicherheitszeit der Steuerung und die möglichen Verzögerungs- und Reaktionszeiten der Abschaltglieder zu berücksichtigen.
Die sich daraus ergebende Gesamtzeit muss kleiner sein als die ***Prozesssicherheitszeit*** (→ Seite 481, "***Die Prozesssicherheitszeit***" → Seite 35) der Anwendung.

PWM

PWM = Puls-Weiten-Modulation
Via PWM kann ein (vom Gerät dazu befähigter) digitaler Ausgang mittels regelmäßiger, schneller Impulse eine beinahe analoge Spannung ausgeben. Bei dem PWM-Ausgangssignal handelt es sich um ein getaktetes Signal zwischen GND und Versorgungsspannung.
Innerhalb einer festen Periode (PWM-Frequenz) wird das Puls-/Pausenverhältnis variiert. Durch die angeschlossene Last stellt sich je nach Puls-/Pausenverhältnis der entsprechende Effektivstrom ein.

R

Ratio

Ratio (lat.) = Verhältnis
Messungen können auch ratiometrisch erfolgen = Verhältnismessung. Das Eingangssignal erzeugt ein Ausgangssignal, das in einem bestimmten Verhältnis zu ihm liegt. Das bedeutet, ohne zusätzliche Referenzspannung können analoge Eingangssignale ausgewertet werden. Ein Schwanken der Versorgungsspannung hat auf diesen Messwert keinen Einfluss.
→ Kapitel ***Bausteine: Zählerfunktionen zur Frequenz- und Periodendauermessung*** (→ Seite 297)

RAW-CAN

RAW-CAN bezeichnet das reine →CAN-Protokoll, das ohne ein zusätzliches Kommunikationsprotokoll auf dem CAN-Bus (auf ISO/OSI-Schicht 2) arbeitet. Das CAN-Protokoll ist international nach →ISO 11898-1 definiert und garantiert zusätzlich in →ISO 16845 die Austauschbarkeit von CAN-Chips.

redundant

Redundanz ist das Vorhandensein von mehr als den notwendigen Mitteln, damit eine Funktionseinheit eine geforderte Funktion ausführt oder damit Daten eine Information darstellen können.

Man unterscheidet verschiedene Arten der Redundanz:

- Die funktionelle Redundanz zielt darauf ab, sicherheitstechnische Systeme mehrfach parallel auszulegen, damit beim →Ausfall einer Komponente die anderen den Dienst gewährleisten.
- Zusätzlich versucht man, die redundanten Systeme voneinander räumlich zu trennen. Dadurch minimiert man das →Risiko, dass sie einer gemeinsamen Störung unterliegen.
- Schließlich verwendet man manchmal Bauteile unterschiedlicher Hersteller, um zu vermeiden, dass ein systematischer Fehler sämtliche redundanten Systeme ausfallen lässt (→diversitäre Redundanz).

Die Software von redundanten Systemen sollte sich möglichst in den folgenden Aspekten unterscheiden:
• Spezifikation (verschiedene Teams),
• Spezifikationssprache,
• Programmierung (verschiedene Teams),
• Programmiersprache,
• Compiler.

remanent

Remanente Daten sind gegen Datenverlust bei Spannungsausfall geschützt.
Z.B. kopiert das →Laufzeitsystem die remanenten Daten automatisch in einen →Flash-Speicher, sobald die Spannungsversorgung unter einen kritischen Wert sinkt. Bei Wiederkehr der Spannungsversorgung lädt das Laufzeitsystem die remanenten Daten zurück in den Arbeitsspeicher.
Dagegen sind die Daten im Arbeitsspeicher einer Steuerung flüchtig und bei Unterbrechung der Spannungsversorgung normalerweise verloren.

Restrisiko

Das ist das verbleibende →Risiko, nachdem →Schutzmaßnahmen ergriffen wurden. Vor dem Restrisiko muss in Betriebsanleitungen und an der Maschine deutlich gewarnt werden.

Risiko

Als Risiko gilt die Kombination der Wahrscheinlichkeit des Eintritts eines →Schadens und des Ausmaßes des Schadens.

Risikoanalyse

Kombination aus ...

- Festlegung der Grenzen der Maschine (Verwendungszweck, zeitliche Grenzen),
- Identifizierung der →Gefährdung (Eingreifen von Personen, Betriebszustände der Maschine, vorhersehbarer Missbrauch) und
- der Risikoeinschätzung (Verletzungsgrad, Schadensumfang, Häufigkeit und Dauer der Gefahr, Eintrittswahrscheinlichkeit, Möglichkeit zur Vermeidung oder Begrenzung des →Schadens).

Risikobeurteilung

Das ist die Gesamtheit des Verfahrens, das die →Risikoanalyse und die →Risikobewertung umfasst. Nach Maschinenrichtlinie 2006/42/EG gilt: "Der Hersteller einer Maschine oder sein Bevollmächtigter hat dafür zu sorgen, dass eine Risikobeurteilung vorgenommen wird, um die für die Maschine geltenden Sicherheits- und Gesundheitsanforderungen zu ermitteln. Die Maschine muss dann unter Berücksichtigung der Ergebnisse der Risikobeurteilung konstruiert und gebaut werden." (→ Anhang 1, Allgemeine Grundsätze)

Risikobewertung

Das ist die auf der →Risikoanalyse beruhende Beurteilung, ob die Ziele zur Risikominderung erreicht wurden.

ro

ro = read only (engl.) = nur lesen
Unidirektionale Datenübertragung: Daten können nur gelesen werden, jedoch nicht verändert.

RTC

RTC = **R**eal **T**ime **C**lock = Echtzeituhr
Liefert (batteriegepuffert) aktuell Datum und Uhrzeit. Häufiger Einsatz beim Speichern von Fehlermeldungsprotokollen.

Rückstellung, manuell

Die manuelle Rückstellung ist eine interne Funktion des →SRP/CS zum anuellen Wiederherstellen einer oder mehrerer →Sicherheitsfunktionen. Wird vor dem Neustart einer Maschine verwendet.

rw

rw = read/write (engl.) = lesen und schreiben
Bidirektionale Datenübertragung: Daten können sowohl gelesen als auch verändert werden.

S

SAE J1939

Das Netzwerkprotokoll SAE J1939 beschreibt die Kommunikation auf einem →CAN-Bus in Nutzfahrzeugen zur Übermittlung von Diagnosedaten (z.B.Motordrehzahl, Temperatur) und Steuerungsinformationen.
Norm: Recommended Practice for a Serial Control and Communications Vehicle Network

- Teil 2: Agricultural and Forestry Off-Road Machinery Control and Communication Network
- Teil 3: On Board Diagnostics Implementation Guide
- Teil 5: Marine Stern Drive and Inboard Spark-Ignition Engine On-Board Diagnostics Implementation Guide
- Teil 11: Physical Layer – 250 kBits/s, Shielded Twisted Pair
- Teil 13: Off-Board Diagnostic Connector
- Teil 15: Reduced Physical Layer, 250 kBits/s, Un-Shielded Twisted Pair (UTP)
- Teil 21: Data Link Layer
- Teil 31: Network Layer
- Teil 71: Vehicle Application Layer
- Teil 73: Application Layer – Diagnostics
- Teil 81: Network Management Protocol

Schaden

Als Schaden bezeichnet man eine physische Verletzung oder Schädigung der Gesundheit.

Schutzmaßnahme

Maßnahme zur vorgesehenen Minderung des →Risikos, z.B.:
- fehlerausschließender Entwurf,
- technische Schutzmaßnahme (trennende Schutzeinrichtung),
- ergänzende Schutzmaßnahme (Benutzerinformation),
- persönliche Schutzausrüstung (Helm, Schutzbrille).

SCT

Bei CANsafety / CANopen Safety überprüft die Sicherheits-Zykluszeit SCT (**S**afeguard **c**ycle **t**ime) die korrekte Funktion der periodischen Übertragung (Daten-Refresh) der →SRDOs. Die Daten müssen innerhalb der eingestellten Zeit wiederholt worden sein, um gültig zu sein. Andernfalls signalisiert die empfangende Steuerung einen schweren Fehler und geht in den sicheren →Zustand (→ Kapitel ***Sicherer Zustand*** (→ Seite 52)).

SD-Card

Eine SD Memory Card (Kurzform für **S**ecure **D**igital Memory Card; deutsch: Sichere digitale Speicherkarte) ist ein digitales Speichermedium, das nach dem Prinzip der →Flash-Speicherung arbeitet.

SDO

SDO = **S**ervice **D**ata **O**bject = Nachrichten-Objekt mit Servicedaten.
Das SDO dient dem Zugriff auf Objekte in einem CANopen-Objektverzeichnis. Dabei fordern 'Clients' die gewünschten Daten von 'Servern' an. Die SDOs bestehen immer aus 8 Bytes.
Beispiele:
- Automatische Konfiguration aller →Slaves über SDOs beim Systemstart.
- Auslesen der Fehlernachrichten aus dem →Objektverzeichnis.

Jedes SDO wird auf Antwort überwacht und wiederholt, wenn sich innerhalb der Überwachungszeit der Slave nicht meldet.

Selbsttest

Testprogramm, das aktiv Komponenten oder Geräte testet. Das Programm wird durch den Anwender gestartet und dauert eine gewisse Zeit. Das Ergebnis davon ist ein Testprotokoll (Log-Datei), aus dem entnommen werden kann, was getestet wurde und ob das Ergebnis positiv oder negativ ist.

Sicherheitsfunktion

Der →Ausfall einer Sicherheitsfunktion einer Maschine kann zum unmittelbar erhöhten →Risiko führen. Der Konstrukteur einer solchen Maschine muss daher:
- einen Ausfall der Sicherheitsfunktion sicher verhindern,
- einen Ausfall der Sicherheitsfunktion rechtzeitig sicher erkennen,
- die Maschine / Anlage bei einem Ausfall der Sicherheitsfunktion rechtzeitig in einen sicheren →Zustand bringen.

Sicherheits-Normentypen

Sicherheitsnormen auf dem Gebiet der Maschinen sind wie folgt strukturiert:
• Typ A-Normen (Sicherheits-Grundnormen)
• Typ B-Normen (Sicherheits-Fachgrundnormen)
• Typ C-Normen (Maschinensicherheitsnormen)

Typ-A-Normen (Sicherheits-Grundnormen) behandeln Grundbegriffe, Entwurfsleitsätze und allgemeine Aspekte, die auf Maschinen angewendet werden können. Beispiele:
• Terminologie, Methodik (ISO 12100),
• Technische Prinzipien (ISO 12100),
• Risikobeurteilung (ISO 12100), ...

Typ-B-Normen (Sicherheits-Fachgrundnormen) behandeln einen Sicherheitsaspekt oder eine Art von Schutzeinrichtungen, die für eine Reihe von Maschinen verwendet werden können. Die Umsetzung auf die konkrete Maschinenfunktion muss bei der Risikobeurteilung, der Spezifikation und der Entwicklung erfolgen. Die Verantwortung für die korrekte Einstufung liegt beim Hersteller der Maschine.

- Typ-B1-Normen für bestimmte Sicherheitsaspekte. Beispiele:
  • Sicherheitsabstände (ISO 13857),
  • Arm-/Hand-Geschwindigkeiten (ISO 13855),
  • Sicherheitsbezogene Teile von Steuerungen (→ISO 13849),
  • Temperaturen, Lärm, ...
- Typ-B2-Normen für Schutzeinrichtungen. Beispiele:
  • NOT-HALT-Schaltungen ((ISO 13850),
  • Zweihand-Schaltungen,
  • trennende oder berührungslos wirkende Schutzeinrichtungen (IEC 61496), ...

Typ-C-Normen (Maschinensicherheitsnormen, Produktnormen) behandeln detaillierte Sicherheitsanforderungen an eine bestimmte Maschine oder eine Gruppe von Maschinen. Oftmals ist dort bereits die Risikoabschätzung für die gesamte Maschine oder Teilfunktionen davon dokumentiert. Bei der Entwicklung einer Maschine sollte man sich deshalb möglichst daran halten. Beispiele:
• Müllfahrzeuge (EN 1501),
• Hubarbeitsbühnen (EN 280), ...

SIL

Der Sicherheits-Integritätslevel SIL ist nach →IEC 62061 eine Einstufung (SIL CL 1...4) der Sicherheitsintegrität der →Sicherheitsfunktionen. Er dient der Beurteilung elektrischer / elektronischer / programmierbar elektronischer (E/E/PE)-Systeme in Bezug auf die Zuverlässigkeit von Sicherheitsfunktionen. Aus dem angestrebten Level ergeben sich die sicherheitsgerichteten Konstruktionsprinzipien, die eingehalten werden müssen, damit das →Risiko einer Fehlfunktion minimiert werden kann.

Slave

Passiver Teilnehmer am Bus, antwortet nur auf Anfrage des →Masters. Slaves haben im Bus eine eindeutige →Adresse.

SRDO

Über SRDOs (**S**afety-**R**elated **D**ata **O**bjects = ***Sicherheitsrelevante Datenobjekte*** (→ Seite 81)) werden bei CANsafety / CANopen Safety sichere Daten ausgetauscht. Ein SRDO besteht immer aus zwei →CAN-Nachrichten mit unterschiedlichen →Identifiern:
• Nachricht 1 enthält die Originalanwenderdaten,
• Nachricht 2 enthält die gleichen Daten, die aber bitweise invertiert werden.

SRP/CS

Safety-**R**elated **P**art of a **C**ontrol **S**ystem = Sicherheitsrelevanter Teil einer Steuerung.
SRP/CS ist ein Teil einer Steuerung, das auf sichere Eingangssignale reagiert und sichere Ausgangssignale erzeugt. Die Kombination sicherheitsrelevanter Teile einer Steuerung beginnt an dem Punkt, an dem sichere Signale erzeugt werden (einschließlich Betätiger z.B. eines Positionsschalters) und endet an den Ausgängen der Leistungssteuerungselemente (einschließlich z.B. der Hauptkontakte eines Schützes).

SRVT

Die sicherheitsrelevante Objekt-Gültigkeitsdauer SRVT (**S**afety-**R**elated **O**bject **V**alidation **T**ime) sorgt bei CANsafety / CANopen Safety dafür, dass die Zeit zwischen den SRDO-Nachrichten-Paaren eingehalten wird:
Nur wenn die redundante, invertierte Nachricht innerhalb der eingestellten Zeit SRVT nach der Original-Nachricht übertragen wurde, sind die damit übertragenen Daten gültig. Andernfalls signalisiert die empfangende Steuerung einen schweren Fehler und geht in den sicheren →Zustand (→ Kapitel ***Sicherer Zustand*** (→ Seite 52)).

Steuerungskonfiguration

Bestandteil der CODESYS-Bedienoberfläche.

► Programmierer teilt dem Programmiersystem mit, welche Hardware programmiert werden soll.

> CODESYS lädt die zugehörigen Bibliotheken.

> Lesen und schreiben der Peripherie-Zustände (Ein-/Ausgänge) ist möglich.

stopped

stopped (engl.) = angehalten
Betriebszustand eines CANopen-Teilnehmers. In diesem Modus werden nur →NMT-Kommandos übertragen.

Symbole

Piktogramme sind bildhafte Symbole, die eine Information durch vereinfachte grafische Darstellung vermitteln (→ Kapitel ***Was bedeuten die Symbole und Formatierungen?*** (→ Seite 9)).

Systemvariable

Variable, auf die via IEC-Adresse oder Symbolname aus der SPS zugegriffen werden kann.

T

Target

Das Target enthält für CODESYS die Hardware-Beschreibung des Zielgeräts, z.B.: Ein- und Ausgänge, Speicher, Dateiablageorte.
Entspricht einem elektronischen Datenblatt.

TCP

Das **T**ransmission **C**ontrol **P**rotocol ist Teil der Protokollfamilie TCP/IP. Jede TCP/IP-Datenverbindung hat einen Sender und einen Empfänger. Dieses Prinzip ist eine verbindungsorientierte Datenübertragung. In der TCP/IP-Protokollfamilie übernimmt TCP als verbindungsorientiertes Protokoll die Aufgabe der Datensicherheit, der Datenflusssteuerung und ergreift Maßnahmen bei einem Datenverlust. (vgl.: →UDP)

Template

Template (englisch = Schablone) ist eine Vorlage, die mit Inhalten gefüllt werden kann.
Hier: Eine Struktur von vorkonfigurierten Software-Elementen als Basis für ein Anwendungsprogramm.

Testrate rt

Die Testrate r_t ist die Häufigkeit der automatischen Tests, um →Fehler in einem →SRP/CS rechtzeitig zu bemerken.

U

Überwachung

Die Überwachung ist eine →Sicherheitsfunktion, die sicherstellt, dass eine →Schutzmaßnahme eingeleitet wird, sobald Folgendes eintritt:
- Die Fähigkeit eines Bauteils oder eines Elements, seine Funktion auszuführen, wird vermindert.
- Die Betriebsbedingungen werden so verändert, dass das resultierende →Risiko steigt.

UDP

UDP (**U**ser **D**atagram **P**rotocol) ist ein minimales, verbindungsloses Netzprotokoll, das zur Transportschicht der Internetprotokollfamilie gehört. Aufgabe von UDP ist es, Daten, die über das Internet übertragen werden, der richtigen Anwendung zukommen zu lassen.
Derzeit sind Netzwerkvariablen auf Basis von →CAN und UDP implementiert. Die Variablenwerte werden dabei auf der Basis von Broadcast-Nachrichten automatisch ausgetauscht. In UDP sind diese als Broadcast-Telegramme realisiert, in CAN als →PDOs.
Dem Protokoll entsprechend, sind diese Dienste nicht bestätigte Dienste: es gibt keine Kontrolle, ob die Nachricht auch beim Empfänger ankommt. Netzwerkvariablen-Austausch entspricht einer "1-zu-n-Verbindung" (1 Sender zu n Empfängern).



V

Verwendung, bestimmungsgemäß

Das ist die Verwendung eines Produkts in Übereinstimmung mit den in der Anleitung bereitgestellten Informationen.



W

Watchdog

Der Begriff Watchdog (englisch; Wachhund) wird verallgemeinert für eine Komponente eines Systems verwendet, die die Funktion anderer Komponenten beobachtet. Wird dabei eine mögliche Fehlfunktionen erkannt, so wird dies entweder signalisiert oder geeignete Programm-Verzweigungen eingeleitet. Das Signal oder die Verzweigungen dienen als Auslöser für andere kooperierende Systemkomponenten, die das Problem lösen sollen.

wo

wo = write only (engl.) = nur schreiben
Unidirektionale Datenübertragung: Daten können nur verändert werden, jedoch nicht gelesen.

Z

Zustand, sicher

Der Zustand einer Maschine gilt als sicher, wenn von ihr keine →Gefährdung mehr ausgeht. Dies ist meist der Fall, wenn alle gefahrbringenden Bewegungsmöglichkeiten abgeschaltet sind und nicht unerwartet wieder anlaufen können.

Zykluszeit

Das ist die Zeit für einen Zyklus. Das SPS-Programm läuft einmal komplett durch.

Je nach ereignisgesteuerten Verzweigungen im Programm kann dies unterschiedlich lange dauern.

10 Index

A

B

C

D

E

F

G

H

T

U

V

W

Z

11 Notizen • Notes • Notes



12 ifm weltweit • ifm worldwide • ifm à l'échelle internationale

Stand: 2014-04-03 8310

www.ifm.com • E-Mail: info@ifm.com

Service-Hotline: 0800 16 16 16 4 (nur Deutschland, Mo...Fr, 07.00...18.00 Uhr)

ifm Niederlassungen • Sales offices • Agences

D	ifm electronic gmbh Vertrieb Deutschland Niederlassung Nord • 31135 Hildesheim • Tel. 0 51 21 / 76 67-0 Niederlassung West • 45128 Essen • Tel. 02 01 / 3 64 75 -0 Niederlassung Mitte-West • 58511 Lüdenscheid • Tel. 0 23 51 / 43 01-0 Niederlassung Süd-West • 64646 Heppenheim • Tel. 0 62 52 / 79 05-0 Niederlassung Baden-Württemberg • 73230 Kirchheim • Tel. 0 70 21 / 80 86-0 Niederlassung Bayern • 82178 Puchheim • Tel. 0 89 / 8 00 91-0 Niederlassung Ost • 07639 Tautenhain • Tel. 0 36 601 / 771-0 ifm electronic gmbh • Friedrichstraße 1 • 45128 Essen
A	ifm electronic gmbh • 1120 Wien • Tel. +43 16 17 45 00
AUS	ifm efector pty ltd. • Mulgrave Vic 3170 • Tel. +61 3 00 365 088
B, L	ifm electronic N.V. • 1731 Zellik • Tel. +32 2 / 4 81 02 20
BR	ifm electronic Ltda. • 03337-000, Sao Paulo SP • Tel. +55 11 / 2672-1730
CH	ifm electronic ag • 4 624 Härkingen • Tel. +41 62 / 388 80 30
CN	ifm electronic (Shanghai) Co. Ltd. • 201203 Shanghai • Tel. +86 21 / 3813 4800
CND	ifm efector Canada inc. • Oakville, Ontario L6K 3V3 • Tel. +1 800-441-8246
CZ	ifm electronic spol. s.r.o. • 25243 Průhonice • Tel. +420 267 990 211
DK	ifm electronic a/s • 2605 BROENDBY • Tel. +45 70 20 11 08
E	ifm electronic s.a. • 08820 El Prat de Llobregat • Tel. +34 93 479 30 80
F	ifm electronic s.a. • 93192 Noisy-le-Grand Cedex • Tél. +33 0820 22 30 01
FIN	ifm electronic oy • 00440 Helsinki • Tel . +358 75 329 5000
GB, IRL	ifm electronic Ltd. • Hampton, Middlesex TW12 2HD • Tel. +44 208 / 213-0000
GR	ifm electronic Monoprosopi E.P.E. • 15125 Amaroussio • Tel. +30 210 / 6180090
H	ifm electronic kft. • 9028 Györ • Tel. +36 96 / 518-397
I	ifm electronic s.a. • 20041 Agrate-Brianza (MI) • Tel. +39 039 / 68.99.982
IL	Astragal Ltd. • Azur 58001 • Tel. +972 3 -559 1660
IND	ifm electronic India Branch Office • Kolhapur, 416234 • Tel. +91 231-267 27 70
J	efector co., ltd. • Chiba-shi, Chiba 261-7118 • Tel. +81 043-299-2070
MAL	ifm electronic Pte. Ltd • 47100 Puchong Selangor • Tel. +603 8063 9522
MEX	ifm efector S. de R. L. de C. V. • Monterrey, N. L. 64630 • Tel. +52 81 8040-3535
N	Sivilingeniør J. F. Knudtzen A/S • 1396 Billingstad • Tel. +47 66 / 98 33 50
NL	ifm electronic b.v. • 3843 GA Harderwijk • Tel. +31 341 / 438 438
P	ifm electronic s.a. • 4430-208 Vila Nova de Gaia • Tel. +351 223 / 71 71 08
PL	ifm electronic Sp. z o.o. • 40-524 Katowice • Tel. +48 32-608 74 54
RA, ROU	ifm electronic s.r.l. • 1107 Buenos Aires • Tel. +54 11 / 5353 3436
ROK	ifm electronic Ltd. • 140-884 Seoul • Tel. +82 2 / 790 5610
RP	Gram Industrial, Inc. • 1770 Mantilupa City • Tel. +63 2 / 850 22 18
RUS	ifm electronic • 105318 Moscow • Tel. +7 495 921-44-14
S	ifm electronic a b • 41250 Göteborg • Tel. +46 31 / 750 23 00
SGP	ifm electronic Pte. Ltd. • Singapore 609 916 • Tel. +65 6562 8661/2/3
SK	ifm electronic s.r.o. • 835 54 Bratislava • Tel. +421 2 / 44 87 23 29
THA	SCM Allianze Co., Ltd. • Bangkok 10 400 • Tel. +66 02 615 4888
TR	ifm electronic Ltd. Sti. • 34381 Sisli/Istanbul • Tel. +90 212 / 210 50 80
UA	TOV ifm electronic • 02660 Kiev • Tel. +380 44 501 8543
USA	ifm efector inc. • Exton, PA 19341 • Tel. +1 610 / 5 24-2000
ZA	ifm electronic (Pty) Ltd. • 0157 Pretoria • Tel. +27 12 345 44 49